Universidad de las Ciencias Informáticas
Facultad 7

Título: Proceso de desarrollo basado en la Arquitectura Orientada a Servicios para el proyecto APS.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Camilo Valentín Estrada
Karelia Valdivia Mola

Tutores: Ing. Sergio Cabrera Sánchez.
Ing. Rosalía Cué Delgado.

Asesor: Lic. Maykell Sánchez Romero.

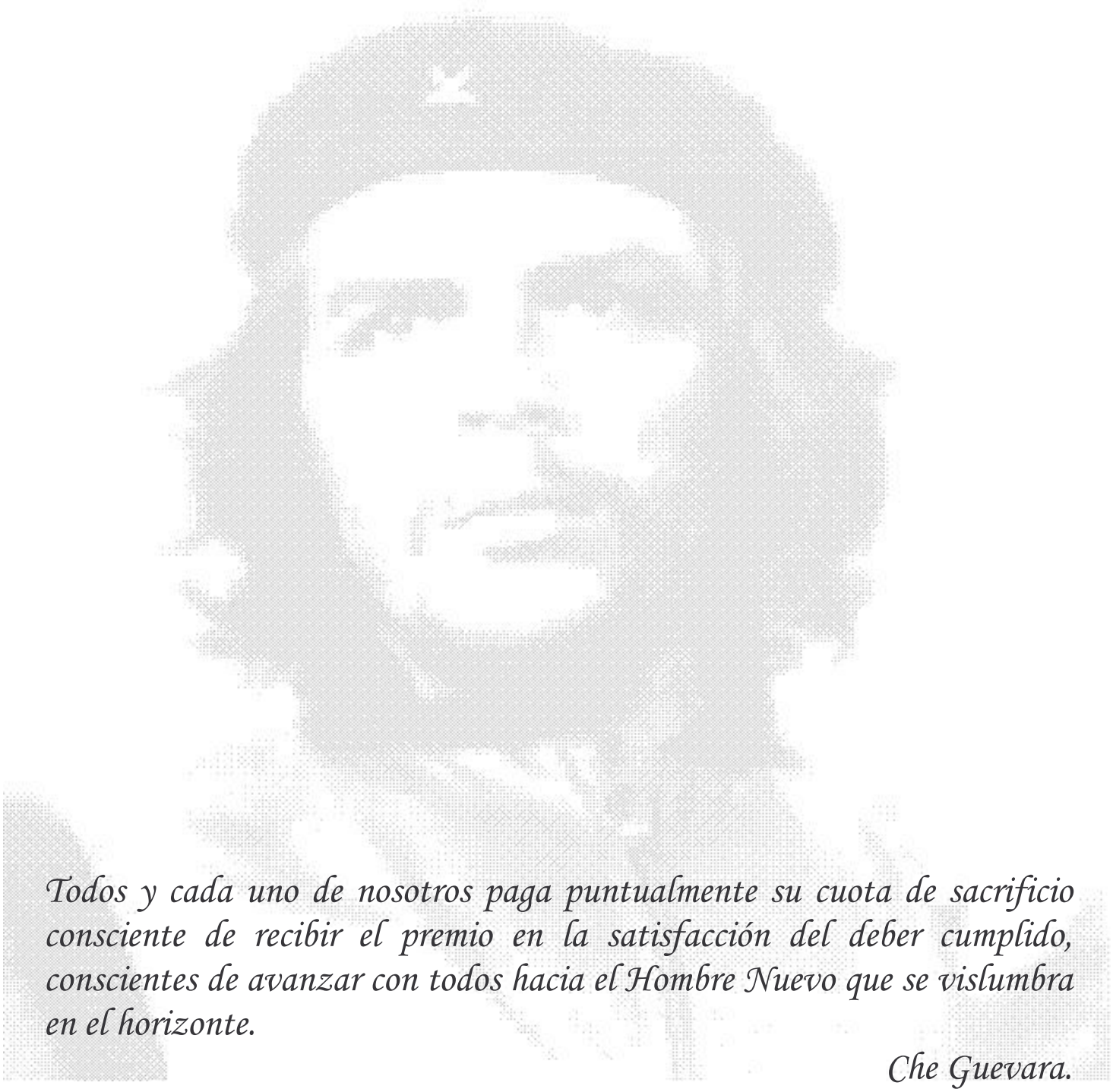
Ciudad de la Habana, junio de 2007.

DECLARACIÓN DE AUTORÍA.

Declaramos que somos los únicos autores de este trabajo en el cual hemos utilizado información y documentación que es propiedad de la empresa SOFTEL lo cual está sujeto a un acuerdo de confidencialidad. Ponemos a disposición de la Universidad de las Ciencias Informáticas (UCI) todo aquello que no comprometa dicho acuerdo.

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

Autores:	Tutores:
Camilo Valentín Estrada.	Ing. Sergio Cabrera Sánchez.
Karelia Valdivia Mola.	Ing. Rosalía Cué Delgado.
_____	_____
_____	_____



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Che Guevara.

DATOS DE CONTACTO

Ing. Sergio Cabrera Sánchez: graduado de Ingeniería SAD, Lvov, Ucrania 1980. Profesor Asistente. Impartió asignaturas de Administración de centros de cálculo, SAD, Cibernética Matemática, Análisis Matemático I y II, Algebra Lineal en la UCLV, Microeconomía y Matemática Financiera en la SUM Marianao. Diplomado en Comercio Exterior, Cursos de Postgrado Comercio Internacional, Asociaciones Económicas Internacionales, Marketing Estratégico, Microeconomía. Jefe Proyecto Medios de Diagnóstico e Imaginología.

Correo electrónico: sergio@softel.cu

Ing. Rosalía L. Cué Delgado: graduada de Ingeniería en Sistema Automatizados en el año 1995. Profesora Auxiliar Adjunto Facultad No. 7 en la disciplina de Ingeniería de Software. Posee 11 años de experiencia en el desarrollo de software desempeñando diferentes roles. En la actualidad se desempeña como Especialista Principal en Ingeniería de Requerimientos de la dirección de desarrollo de la empresa SOFTEL.

Correo electrónico: rosalia@softel.cu

Lic. Maykell Sánchez Romero: graduado de Licenciatura en Ciencias de la Computación de la Universidad de La Habana, profesor instructor. Ha sido Jefe de Departamento de Ciencias de la Especialidad de la Facultad 7 de la UCI, arquitecto de la Facultad 7, Jefe de Grupo de Investigación y Jefe de Proyecto.

Correo electrónico: maykell@uci.cu

AGRADECIMIENTOS

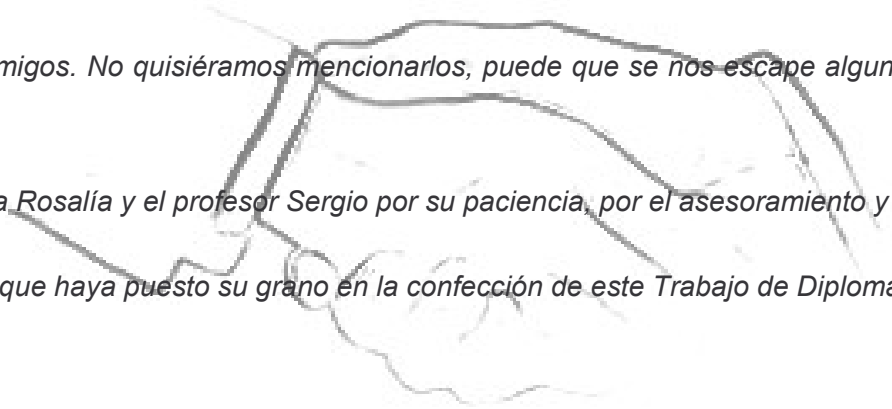
Queremos agradecer en primer lugar la oportunidad de cursar estudios en la primera Universidad surgida al calor de la Batalla de Ideas impulsada por nuestro Comandante en Jefe y por el privilegio de integrar el Destacamento "Tropa Futuro". Gracias a la Revolución por las múltiples oportunidades para superarnos.

A nuestros padres, por el apoyo, la confianza y el amor, por sus regaños y consejos, por ser nuestros principales guías y por ayudarnos a llegar hasta aquí.

A nuestros amigos. No quisiéramos mencionarlos, puede que se nos escape alguno, ellos saben quienes son.

A la profesora Rosalía y el profesor Sergio por su paciencia, por el asesoramiento y ayuda incondicional.

A todo aquel que haya puesto su grano en la confección de este Trabajo de Diploma.



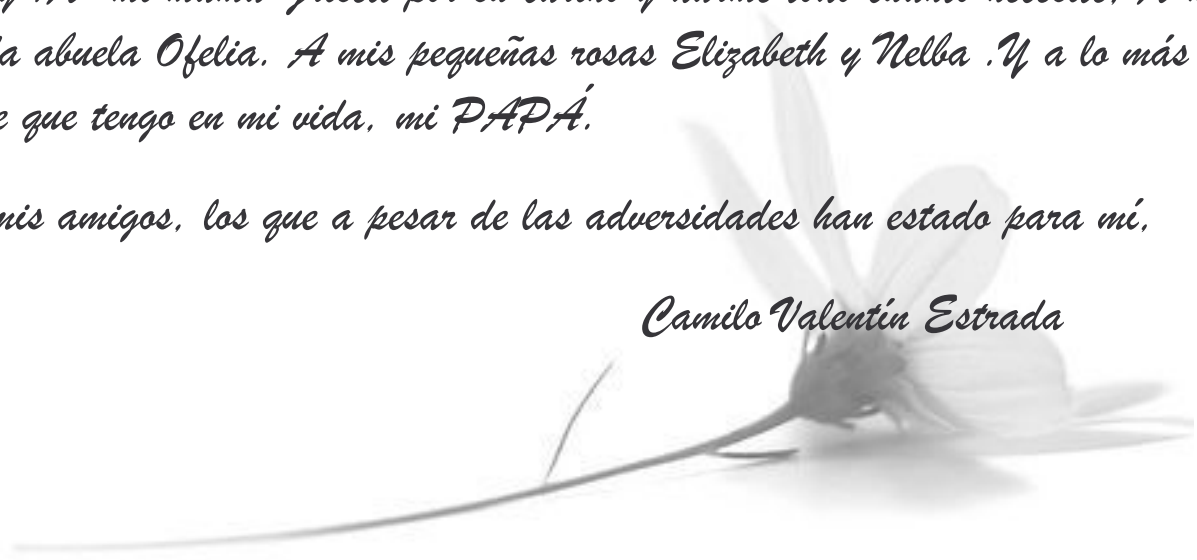
DEDICATORIA

A mi MAMÁ donde quiera que esté.

A toda mi familia, en especial, a mis tías y a mis abuelitas de Campo Hatuey .A mi mamá Grisell por su cariño y darme todo cuanto necesité. A mi querida abuela Ofelia. A mis pequeñas rosas Elizabeth y Nelba .Y a lo más grande que tengo en mi vida, mi PAPÁ.

A mis amigos, los que a pesar de las adversidades han estado para mí.

Camilo Valentín Estrada



A toda mi familia, en especial a mis padres, mi abuela Delia, mis hermanos y mis tías Mayda y Diltma por su cariño y apoyo incondicional.

A mis amistades que me apoyaron en todo momento, a los distantes, a los cercanos, a los de siempre.

Karelia Valdivia Mola

RESUMEN

El presente trabajo de diploma, tiene como objetivo obtener un procedimiento para el desarrollo de las aplicaciones sobre Arquitectura Orientada a Servicios en el proyecto Atención Primaria de Salud (APS), siguiendo las extensiones de la metodología Rational Unified Process (RUP). Se incorporan nuevas actividades y artefactos que se generan durante la disciplina de Análisis y Diseño. Además de la inclusión, de nuevas responsabilidades encomendadas a los roles de diseñador y arquitecto del software para la creación de un Modelo de Servicios. Proporcionando guías de trabajo para una mejor comprensión del proceso desarrollado.

La configuración del proceso en el proyecto APS, fue realizada a partir del Plug-in SOA. Este constituye, una actualización para el proceso de desarrollo de Rational, a utilizarse en aplicaciones con Arquitectura Orientada a Servicios. Además, se incluye una breve descripción de la herramienta Rational Software Architect, recomendada para ser utilizada en la ejecución de este proceso de desarrollo.

Esta investigación permitirá contar con un documento normativo que resuma la configuración de un proceso de desarrollo de aplicaciones con arquitectura SOA para el proyecto APS y la empresa SOFTEL, obteniendo una guía para ejecutar dicho proceso. Proveerá una mayor flexibilidad a la infraestructura de la empresa y una mayor simplicidad en la corriente de información tanto interna como externa, acoplando los sistemas de manera más rápida. Permitirá enfrentarse a los desafíos de reducción de costos, heterogeneidad, flexibilidad frente al cambio, tiempo de puesta en producción, integración y reusabilidad. Todo esto traerá consigo una mayor productividad en el proyecto.

Palabras Claves:

Rational Unified Process (RUP)**,** proceso de desarrollo, Arquitectura Orientada a Servicios (SOA)**,** Atención Primaria de Salud (APS), Análisis y Diseño, Modelo de Servicios.

TABLA DE CONTENIDOS.

LISTA DE ILUSTRACIONES.

LISTA DE TABLAS.

LISTA DE ANEXOS.

INTRODUCCIÓN.

A lo largo de los últimos años con el desarrollo de las tecnologías de la información surge el modelo de computación distribuida, el cual centra sus esfuerzos en el deseo de resolver diferentes problemas asociados con la administración distribuida de los sistemas cliente - servidor.

La computación distribuida permite que diferentes aplicaciones conversen unas con otras aún cuando no se encuentran en la misma computadora, estos sistemas están soportados por una capa intermedia (middleware), o más específicamente por una capa de mensajes que conecta al cliente con los datos.[1]

Las aplicaciones del Proyecto de Atención Primaria de Salud (APS) atendido por la empresa de producción de software SOFTEL se basan en la Arquitectura Basada en Componentes y la orientada a servicios (SOA), y su proceso de desarrollo está basado en la metodología RUP, la cual solo contempla la Arquitectura Basada en Componentes. Lo cierto es, que no está definido un proceso de desarrollo para aplicaciones cuya arquitectura es SOA y no existe actualmente una guía o documento que resuma dicho proceso para cada una de sus fases y que estereotipos y artefactos se utilizan y el rol que desempeñan.

La **Arquitectura Orientada a Servicios** (en inglés Service Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de los servicios para dar soporte a los requerimientos de software del usuario y expone los procesos de negocio como servicios, propiedad esta, que define el carácter flexible de esta arquitectura. [2]

Con el auge y el desarrollo alcanzado en los últimos años por la Arquitectura Orientada a Servicios su empresa líder en el mercado IBM (**I**nternational **B**usiness **M**achines), con una inversión de alrededor de mil millones de dólares lanzó a finales del año 2005 como parte de una operación comercial el plug-in SOA, el cual constituye una actualización para el Rational Unified Process (RUP) donde se describen nuevas actividades y artefactos que pasan a formar parte del proceso de desarrollo de Rational y se introduce además una guía para el arquitecto y el diseñador de software para ejecutar un proceso de desarrollo utilizando la Arquitectura Orientada a Servicios.

---

[1] GARCIA, A. y A. PANEQUE. *Patrones Arquitectónicos en las Aplicaciones del Proyecto Atención Primaria de la Salud (APS)*, 2006.
[2] ALCUBILLA, J. C. *ABC...SOA...Primera Parte*, 2007. Disponible en: http://www.tecnologiahechapalabra.com/datos/soluciones/gerencia/articulo.asp?i=960

Este plug-in desarrollado por la trasnacional IBM, constituye otra experiencia en el desarrollo de la Arquitectura Orientada a Servicios. En él se describen e introducen nuevas pautas y prácticas como parte de esta nueva capacidad incorporada a la Suite de Rational.

Básicamente el proceso de desarrollo tiene que ayudar a construir software, por ello pone las reglas necesarias para alcanzar con éxito el propósito de toda empresa productora de software: desarrollar un producto de software en tiempo, dentro del presupuesto fijado y que cumpla las expectativas y necesidades de los clientes, dejando la libertad suficiente para no sentirse agobiados.

Con el presente trabajo se propone definir un proceso de desarrollo basado en RUP para aplicaciones cuya arquitectura es SOA.

Situación Problémica.

Las aplicaciones del Proyecto de Atención Primaria de Salud afrontan actualmente varios problemas que debido a la dinámica del proceso de desarrollo no ha permitido a los especialistas del proyecto detenerse en la investigación de estándares que brinden mejores prácticas para el modelado de aplicaciones cuya arquitectura sea la orientada a servicios.

Para aplicaciones con Arquitectura Orientada a Servicios actualmente el proyecto no cuenta con un documento normativo que resuma:

- ✓ Para cada fase del proceso y en cada disciplina:
    - Roles y responsabilidades.
    - Actividades a realizar.
    - Artefactos y estereotipos a generar.

Por tanto el **problema** principal consiste en la necesidad de disponer de un proceso de desarrollo para aplicaciones cuya arquitectura sea SOA.

El **objeto de estudio** es configurar un proceso de desarrollo de aplicaciones con arquitectura SOA para el proyecto de Atención Primaria de Salud a partir del plug-in SOA para RUP.

De ello se deriva que el **campo de acción** que comprende este trabajo, es el proceso de desarrollo de aplicaciones con arquitectura SOA siguiendo la metodología RUP en el proyecto APS.

Si hipotéticamente se contara con un documento estándar para el proceso de desarrollo de aplicaciones con arquitectura SOA; que reúna artefactos y estereotipos con los roles que cada uno implementa, se ofrecería una mayor asistencia a los especialistas, facilitaría el modelado y la implementación de estas aplicaciones sobre arquitectura SOA, logrando mejores prácticas de industria, arquitecturas de referencia, modelos y componentes de software para agilizar el proceso de diseño, y reducir costos y riesgos de los proyectos de implementación de las aplicaciones en el proyecto APS.

El **objetivo general** de esta investigación será obtener un procedimiento para el desarrollo de las aplicaciones sobre Arquitectura Orientada a Servicios en el proyecto APS siguiendo las extensiones de la metodología RUP.

A partir de este objetivo global surgen **objetivos** más **específicos:**

- ✓ Definición del proceso de desarrollo para las aplicaciones del proyecto APS cuya arquitectura es SOA.

Para alcanzar los objetivos y solucionar la situación problémica anteriormente planteada, es necesario la realización de las siguientes **tareas**:

- ✓ Examinar el plug-in de RUP para SOA.
- ✓ Describir el proceso, lo cual incluye:
    - Fases y disciplinas.
    - Actividades de cada fase y disciplinas.

- Artefactos de cada actividad.
- Trabajadores y responsabilidades dentro de cada fase y disciplina.

✓ Configurar el proceso de desarrollo de RUP para SOA en las aplicaciones del proyecto APS.

El presente documento de diploma consta de Resumen, listado de Tablas y Anexos, Introducción, tres capítulos, seguido de Conclusiones, Recomendaciones, Referencias Bibliograficas, Bibliografía, Anexos y el Glosario de Términos.

El capítulo 1 brinda una pequeña panorámica del desarrollo alcanzado por otras metodologías en el trabajo con SOA y las características de cada una de ellas. También se describe brevemente el contenido del plug- in de RUP para SOA y las disciplinas que afecta.

El capítulo 2 muestra una descripción detallada del plug-in con todos los cambios que trae consigo: la inclusión de nuevas actividades y artefactos que forman parte del nuevo proceso de desarrollo de RUP para aplicaciones con Arquitectura Orientada a Servicios.

En el capítulo 3 se configura el proceso de desarrollo de RUP para la Arquitectura Orientada a Servicios en las aplicaciones del proyecto APS describiendo los nuevos conceptos a tener en cuenta para cada una de las fases y disciplinas, además de las nuevas actividades y artefactos.

El capítulo 4 está compuesto por varias guías de trabajo necesarias para la utilización y comprensión de los nuevos artefactos y actividades que se adicionan.

En las Conclusiones se da respuesta a cada una de las tareas de investigación planteadas al inicio del trabajo.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA.

Este capítulo ofrece un enfoque general de los aspectos relacionados a la existencia y experiencias de otras metodologías de desarrollo con relación a la Arquitectura Orientada a Servicios. Brinda una pequeña descripción de estas metodologías seleccionadas de acuerdo a su popularidad y usabilidad a nivel mundial, así como la herramienta escogida para ejecutar el proceso de desarrollo para SOA.

Se realiza una pequeña introducción a la funcionalidad que muestra el plug-in de RUP para SOA. Además de una breve descripción de las disciplinas que afecta y las nuevas actividades y responsabilidades que se asignan al diseñador y al arquitecto del software, así como los artefactos que se generan.

1.1 Metodologías de desarrollo.

Hoy en día la metodología RUP a nivel mundial no se encuentra sola en cuanto al trabajo con la Arquitectura Orientada a Servicios se refiere. Existen otras metodologías de desarrollo como FDD (Desarrollo Manejado por Rasgos), MSF (Marco de Soluciones de Microsoft), MDA (Arquitectura Dirigida por Modelos) y XP (e**X**treme **P**rogramming) que están comenzando a dar sus primeros pasos en el trabajo con SOA pero ninguna de ellas ha definido como tal un proceso de desarrollo para aplicaciones con Arquitectura Orientada a Servicios.

1.1.1 Feature Driven Development (FDD).

La metodología Desarrollo Manejado por Rasgos (traducción al español) es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca. Consta de 5 fases fundamentales: desarrollo de un modelo general, construcción de la lista de funcionalidades, plan de liberaciones, diseño e implementación, donde todas las fases se planifican, diseñan e implementan en base a las funcionalidades. Se podría considerar a medio camino entre RUP y XP. En estos momentos de adentra en el mundo de SOA tomando su flexibilidad a la hora de brindar servicios pero sin un proceso de desarrollo definido.

1.1.2 Microsoft Solution Framework (MSF).

Es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. El Marco de Soluciones de Microsoft (traducción al español) se centra en los modelos de proceso y de equipo, dejando en un segundo plano las elecciones tecnológicas. MSF tampoco cuenta con una guía para el trabajo con SOA ni define artefactos o estereotipos para el modelado de aplicaciones con Arquitectura Orientada a Servicios.

1.1.3 Model-Driven Architecture (MDA).

Es una marca con licencia de OMG (**O**bject **M**anagement **G**roup) que se basa en el uso de UML para el desarrollo controlado por modelos, este es un concepto promovido (pero no creado) por la OMG, que propone basar el desarrollo de software en modelos especificados utilizando UML, para que a partir de esos modelos, se realicen trasformaciones que generen código u otro modelo, con características de una tecnología particular (o con menor nivel de abstracción). MDA define un framework para procesar y relacionar modelos, pero no tiene un proceso de desarrollo definido y por tanto tampoco para SOA.

1.1.4 eXtreme Programming (XP).

Después de RUP la metodología más conocida y utilizada es la **eXtreme Programming (XP)**, la misma es exitosa porque enfatiza en la satisfacción del cliente y promueve el trabajo en equipo, además elimina las actividades no necesarias en determinados momentos del desarrollo para reducir costos y frustraciones, pero solo es aplicable a proyectos pequeños que solo cuentan con un equipo de desarrolladores compuesto como máximo por 10 integrantes.

1.2 SOA para aplicaciones del proyecto APS.

Como parte del proceso de informatización de la sociedad cubana, la Facultad 7 de la Universidad de las Ciencias Informáticas (UCI) en conjunto con la empresa de producción de software SOFTEL ha venido desarrollando una serie de proyectos con el propósito de informatizar al Sistema Nacional de Salud.

Las aplicaciones del Proyecto Atención Primaria de Salud tienen Arquitectura Basada en Componentes y orientada a servicios (SOA), pero no se cuenta como tal con un proceso de desarrollo que defina las

fases, disciplinas, roles, actividades, y los artefactos que se derivan así como los estereotipos para el desarrollo orientado a servicios.

El proceso de desarrollo de la empresa SOFTEL está basado en la metodología RUP la cual contempla el desarrollo basado en componentes por lo que nuestra investigación se basa en la definición de una guía para el trabajo con SOA a través de RUP.

1.3 Antecedentes de Plug-In SOA de IBM.

Ante el auge de la arquitectura del siglo XXI, la empresa líder en el mercado en su uso, así como en inversión creó un plug-in que reflejara una perspectiva de SOA para RUP dado que esta metodología es las más utilizada en la actualidad por su capacidad de organización para el modelado de todo tipo de proyectos: pequeños, medianos y grandes. Dicho plug-in permite organizar el modelado y documentación de la arquitectura y diseño en aplicaciones con Arquitectura Orientada a Servicios.

1.4 Conceptos básicos.

Modelo de Servicios.

Es la base de una Arquitectura Orientada a Servicios. Se utiliza como entrada esencial a las actividades en las disciplinas de implementación y prueba. Está compuesto por un conjunto de elementos como servicios, proveedores, especificaciones, canales, entradas, mensajes, particiones y las colaboraciones entre ellos.[3]

Servicio.

Un servicio es un elemento del Modelo de Servicios el cual es la base de una Arquitectura Orientada Servicio. Es proporcionado por un proveedor de servicios y posee una especificación del servicio.[4]

Proveedor de Servicios.

[3] JOHNSTON, S. y J. SMITH. *RUP Update for Service-Oriented Architecture* IBM Corporation, 2005.
[4] Igual a Referencia 3.

Un Proveedor de Servicios agrupa un sistema relacionado de servicios.

Especificación del Servicio.

Una Especificación del Servicio es un elemento del Modelo de Servicios que proporciona la especificación estructural y de comportamiento para un caso específico del servicio. Una Especificación del Servicio también puede identificar un sistema de políticas que gobiernan el acceso al servicio o el uso del mismo.[5]

Canal de Servicios.

Un Canal de Servicios es un elemento del Modelo de Servicios que representa una conexión entre dos servicios, o entre un cliente y un servicio. Es importante entender que el canal no representa ninguna interacción en particular.[6]

Mensaje.

Un mensaje es un contenedor de información que identifica un subconjunto del modelo de información o del modelo de dominio, el cual es invocado dentro o fuera del servicio. Un mensaje no debe seguir ningún comportamiento definido.[7]

Colaboración de Servicio.

La Colaboración de Servicio es una representación de un sistema de comunicación entre dos o más servicios encapsulados usualmente como un nuevo servicio. De esta manera el modelo puede representar los servicios cuya implementación es simplemente la colaboración de un sistema de servicios existentes.[8]

Partición de Servicios.

Una Partición de Servicios es un elemento del modelo que proporciona un agrupamiento lógico para los proveedores de servicios, lógico en el sentido que la estructura de partición puede reflejar una estructura

[5] Igual a Referencia 3.
[6] Igual a Referencia 3.
[7] Igual a Referencia 3.
[8] Igual a Referencia 3.

del sistema que afecta la manera en que se despliega el sistema físico o puede representar una estructura que no tenga ningún impacto en el despliegue.[9]

1.5 Plug-In SOA para RUP.

Toda definición de un proceso de desarrollo implica la selección de un conjunto de premisas a partir de las cuales se definen los procesos, tal es el caso de RUP, el cual, entre otras premisas define su proceso sobre la Arquitectura Basada en Componentes. Con el surgimiento de SOA y su gran ventaja de promover la integridad de diferentes aplicaciones, RUP ha definido su proceso para este tipo de arquitectura. Este plug-in es una actualización de Proceso Unificado de Rational para la Arquitectura Orientada a Servicios (SOA) y la implementación del profile de UML en el Rational Software Architect para los servicios de software, brindando tanto métodos del modelado como un conjunto de buenas prácticas para la arquitectura y el diseño de soluciones basadas en el modelo de Arquitectura Orientada a Servicios.

1.5.1 Descripción de disciplinas.

Una disciplina proporciona una visión en los elementos de proceso dentro de RUP desde la perspectiva de las habilidades generales necesarias para su realización. Cada una de ellas describe un conjunto de actividades relacionadas y los artefactos basados alrededor de una habilidad común.

Cada disciplina se describe como sigue en la Tabla 1:

Introducción.	
Conceptos.	
Flujo de Trabajo.	
Descripción de la actividad.	
Descripción del artefacto.	
Descripción de las guías.	

[9] Igual a Referencia 3.

Tabla1: Descripción de las disciplinas

1.5.2 Roles.

Un rol define el comportamiento y responsabilidades de un individuo, o grupo de individuos.[10]

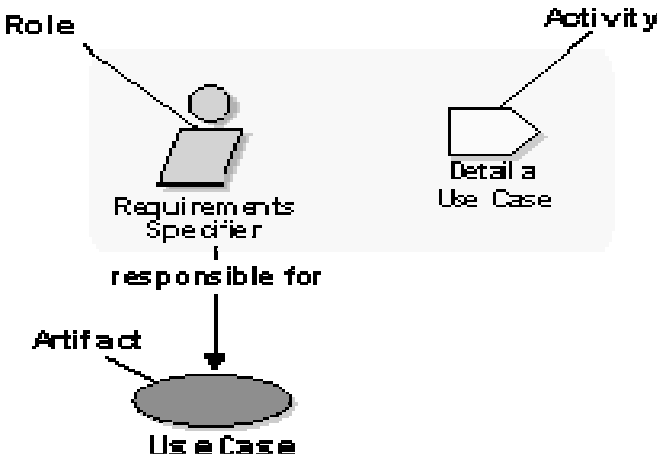


Ilustración 1: Definición de Rol

En el plug-in no se adicionan nuevos roles si no que las nuevas actividades y artefactos que se agregan son realizadas por el arquitecto y el diseñador del software, como se muestra en la Tabla 2.

Rol	Nuevas Actividades	Nuevos Artefactos
Arquitecto de Software	Identificar Servicios	Modelo de Diseño
Diseñador	Diseño de Servicios	Modelo de Servicios

Tabla 2: Novedades del Plug-in

1.5.3 Artefactos.

Los artefactos son resultados tangibles durante el desarrollo de un proyecto. Son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables, documentos, etc.[11]

[10] *INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE*. Curso 2005-2006. [Disponible en: http://teleformacion.uci.cu/mod/resource/view.php?id=6655

El nuevo artefacto que se genera en el flujo de trabajo recibe el nombre de Modelo de Servicios.

El Modelo de Servicios cuenta con una serie de servicios que pueden llevarse a cabo de manera opcional u obligatoria generando tres tipos de reportes como parte de la documentación del proceso que se desarrolla. Está compuesto a su vez por 8 artefactos:

- ✓ Message (Mensaje).
- ✓ Service (Servicio).
- ✓ Service Channel (Canal de Servicios).
- ✓ Service Collaboration (Colaboración de Servicio).
- ✓ Service Gateway (Entrada de Servicio).
- ✓ Service Partition (Partición de Servicios).
- ✓ Service Provider (Proveedor de Servicios).
- ✓ Service Especificación (Especificación del Servicio).

El Modelo de Diseño es un modelo de objetos que describe la realización de los Casos de Uso (CU) del diseño, y sirve como abstracción del modelo de implementación y de su código fuente. Genera cuatro plantillas y reportes: Descripción del Modelo de Diseño, Realización de los CU, Subsistema de Paquetes de Diseño y Reporte de las Clases. En esta nueva actualización de la Suite de Rational se adiciona dentro de este modelo un nuevo artefacto que recibe el nombre de Componente de Servicios.

1.6 Herramientas.

1.6.1 Rational Software Architect.

Es una herramienta de construcción y diseño que potencia el desarrollo gestionado por modelos con UML 2.0 para crear aplicaciones bien estructuradas, incluyendo las basadas en una Arquitectura Orientada a Servicios.

Unifica todos los aspectos del diseño y desarrollo de software en una única herramienta fácil y potente. Soporta análisis, diseño, manejo y evolución de soluciones empresariales y de los servicios.

[11] Igual a Referencia 10.

RSA para el proceso de Rational basado en componentes y en la creación del Modelo de Servicios.

Para desarrollar software correctamente, se necesita un buen proceso y buenas herramientas. El Proceso Unificado del Rational (RUP) se integra con Rational Software Architect y proporciona las directrices y herramientas para aplicar las recomendaciones durante todas las fases del ciclo de vida del proyecto.

Esta herramienta describe los pasos para crear un Modelo de Servicios usando la Arquitectura de Software de Rational. Además se utiliza el perfil de UML para los servicios del software. El Modelo de Servicios descrito durante este proceso de desarrollo para Arquitectura Orientada a Servicios no puede desarrollarse en versiones anteriores de suite de Rational.

El Rational Software Architect permite modelar todas las operaciones contenidas en la Suite de Rational 2003. Dentro de los anexos de esta investigación se incluyen varios diagramas del proceso de desarrollo de Rational basado en componentes utilizando esta herramienta:

- ✓ Diagrama de Casos de Uso.
- ✓ Diagrama de Clases del Análisis.
- ✓ Diagrama de Secuencia.
- ✓ Diagrama de Clases del Diseño.
- ✓ Diagrama de Componentes.

Conclusiones.

En este capitulo se investigó acerca de las diferentes metodologías de desarrollo que existen a nivel mundial, de su campo de trabajo en el empleo de la Arquitectura Orientada a Servicios y de sus características y conceptos esenciales. Además, de investigar que guía de trabajo utilizaban a la hora de emplear SOA y si contaban con un proceso de desarrollo definido para trabajar con esta arquitectura y que pudiera dar solución a nuestro problema.

Se llegó a la conclusión que ninguna de estas metodologías, cuenta con un proceso de desarrollo para aplicaciones con Arquitectura Orientada a Servicios por lo que se hizo necesario el análisis y el estudio del plug-in SOA de IBM para RUP y a partir de él configurar un proceso que satisfaga las necesidades de

desarrollo de las aplicaciones del proyecto APS. Además, se brindó una pequeña introducción sobre el contenido del plug-in y los principales conceptos que incluye.

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA.

En este capítulo se realiza un análisis detallado de las funcionalidades del plug-in junto con una descripción bien profunda de la mayor parte de su contenido con cada uno de los cambios que implica en el análisis y el diseño con la creación de un Modelo de Servicios que incluye sus propios artefactos y estereotipos.

2.1 Descripción Plug-in SOA de IBM para RUP.

El plug-in SOA constituye una actualización para el Rational Unified Process (RUP). Es un intento para describir la conexión que existe entre el Modelo de Negocio y el Modelo de Servicios; muchos proyectos de SOA utilizan modelos de proceso del negocio para entender su negocio, requisitos funcionales y los servicios requeridos para apoyar un proceso.

Alcance.

- ✓ En este plug-in se describen nuevas actividades y artefactos que forman parte del Modelo de Servicios y se adicionan dentro de RUP.
- ✓ Esta nueva capacidad de RUP implica cambios en el análisis y el diseño, creando un Modelo de Servicios compuesto por sus propios artefactos y estereotipos.
- ✓ Se trata de actualizar el RUP existente con las nuevas características que implica el uso de SOA. Significa que no solamente se agregan nuevos conceptos sino que además se describen nuevas actividades y artefactos que se adicionan a los que ya existían en RUP.
- ✓ Para la creación de este plug-in se estudió a fondo la implicación de la Arquitectura Orientada a Servicios en el Modelo del Negocio y su impacto en el diseño del mismo, así como los cambios que implica en el análisis y el diseño, en la organización operacional y la integración del negocio. También se analizaron las diferentes tendencias que muestra SOA para realizar la implementación, el despliegue y la gerencia operacional. El plug-in se centra en ediciones de la arquitectura y del diseño.

La creación de este plug-in fue parte de la operación comercial que se realizó para el lanzamiento de RUP con esta nueva actualización descrita y empaquetada en este plug-in SOA de IBM para Suite de Rational.

El objetivo de esta actualización es brindar una guía específica en el desarrollo de los modelos de arquitectura y de diseño para una aplicación con Arquitectura Orientada a Servicios. Muchas áreas de RUP no han sido afectadas por esta actualización por lo que solo se describen a continuación las afectadas.

2.2 Disciplina Análisis y Diseño.

Los cambios que presenta esta nueva funcionalidad incorporada a la Suite de Rational para el trabajo con SOA en un intento por describir la conexión que existe entre el Modelo de Negocio y el Modelo de Servicio afecta la disciplina de Análisis y Diseño y los detalles de su flujo de trabajo que incluye el Refinamiento de la Arquitectura y el Diseño de Servicios.

Flujos de trabajo de esta disciplina.

- ✓ Definir la Arquitectura Candidata.
- ✓ Realizar la Síntesis Arquitectónica
- ✓ Refinar la Arquitectura.
- ✓ Diseño de la BD.
- ✓ Analizar el Comportamiento
- ✓ Componentes de Diseño.
- ✓ Diseño de Servicios.

En la fase de Inicio, el análisis y el diseño se refieren a establecer si el sistema previsto es factible, y a determinar las tecnologías potenciales para la solución del problema (realizar la síntesis arquitectónica). Si existen factores de riesgo para realizar el proyecto, si se puede modelar bien el negocio e identificar bien actores y CU.

Los objetivos primarios de la fase de elaboración consisten en crear una arquitectura inicial para el sistema (definir una arquitectura candidata) para proporcionar un punto de partida para el trabajo en el análisis. Si ya existe la arquitectura (porque fue producida en iteraciones anteriores, en proyectos anteriores, o se obtuvo de un marco de uso similar al nuevo proyecto que se realiza), el objetivo de trabajo

cambia a refinar la arquitectura y analizar su comportamiento, y a crear un sistema inicial de los elementos que proporcionan un comportamiento apropiado para el proyecto.

Los componentes de diseño proporcionan un comportamiento apropiado para satisfacer los requisitos del sistema. Si el sistema incluye una base de datos, entonces el diseño de la base de datos ocurre en paralelo. El resultado es una propuesta inicial del sistema la cual se refina más a fondo en la fase de implementación.

Refinar la Arquitectura.

Descripción.

Este flujo de trabajo:

- ✓ Proporciona la transición natural de las actividades del análisis a las actividades del diseño, identificando:
    - los elementos del diseño apropiados a partir de los elementos del análisis.
    - los mecanismos del diseño apropiados a partir de los mecanismos relacionados del análisis.
- ✓ Describe la organización del sistema de compilación de la arquitectura de despliegue.
- ✓ Organiza el Modelo de Implementación para hacer la transición entre el diseño y la implementación.
- ✓ Mantiene la consistencia y la integridad de la arquitectura, asegurando:
    - Que los nuevos elementos del diseño identificados para la iteración actual, se integren con los elementos pre existentes del diseño.

Diseño de Servicios.

El propósito de este Flujo de Trabajo es propiciar especificaciones detalladas del comportamiento del servicio y modelar la Carpeta de Servicios en términos de proveedores y particiones de servicios, como se muestra en la Ilustración 2.

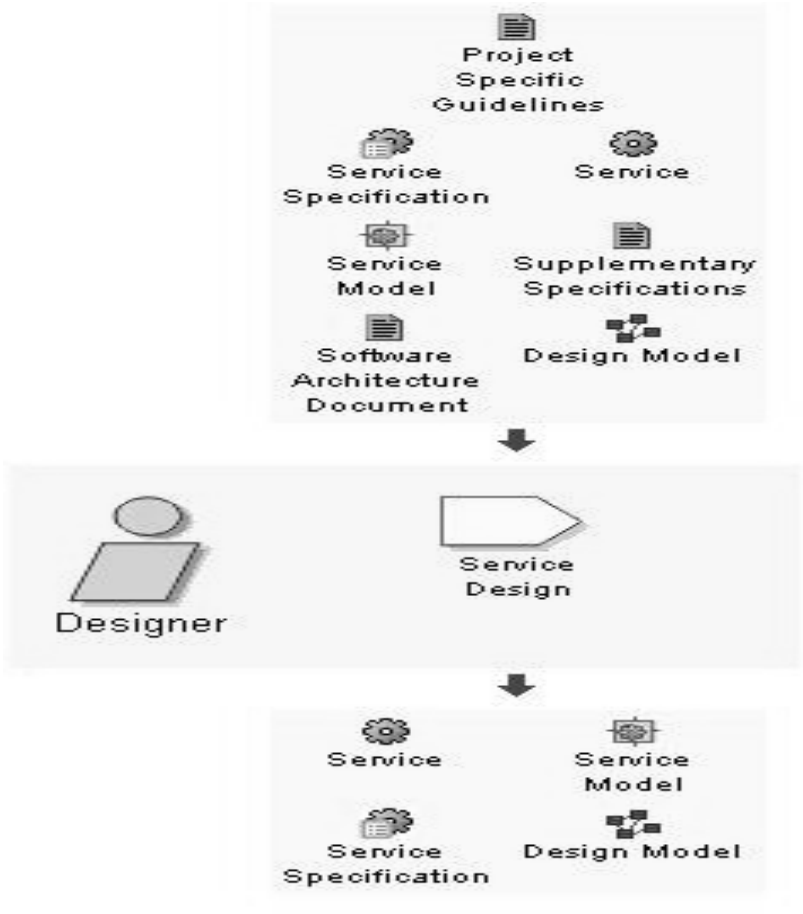


Ilustración 2: Descripción del Diseño de Servicios.

Este detalle del flujo de trabajo persigue los siguientes objetivos:

- ✓ Refinar las definiciones de los elementos de diseño del servicio proporcionado, así como las especificaciones detalladas del comportamiento del servicio y modelar la Carpeta de Servicios en términos de proveedores y particiones de servicio.

- ✓ Refinar y poner al día las realizaciones de los casos de uso basadas en el nuevo elemento de diseño del servicio identificado.
- ✓ Repasar como se desarrolla el diseño.

2.3 Roles.

En esta actualización de RUP no se adicionaron nuevos roles pues el arquitecto de software y el diseñador pueden cumplir satisfactoriamente con los requerimientos asumiendo las nuevas actividades y responsabilidades en la creación de los nuevos artefactos.

2.3.1 Arquitecto de software.

El arquitecto es el responsable de la arquitectura de la aplicación, que incluye las decisiones técnicas que obligan al diseño total y la implementación para el proyecto.

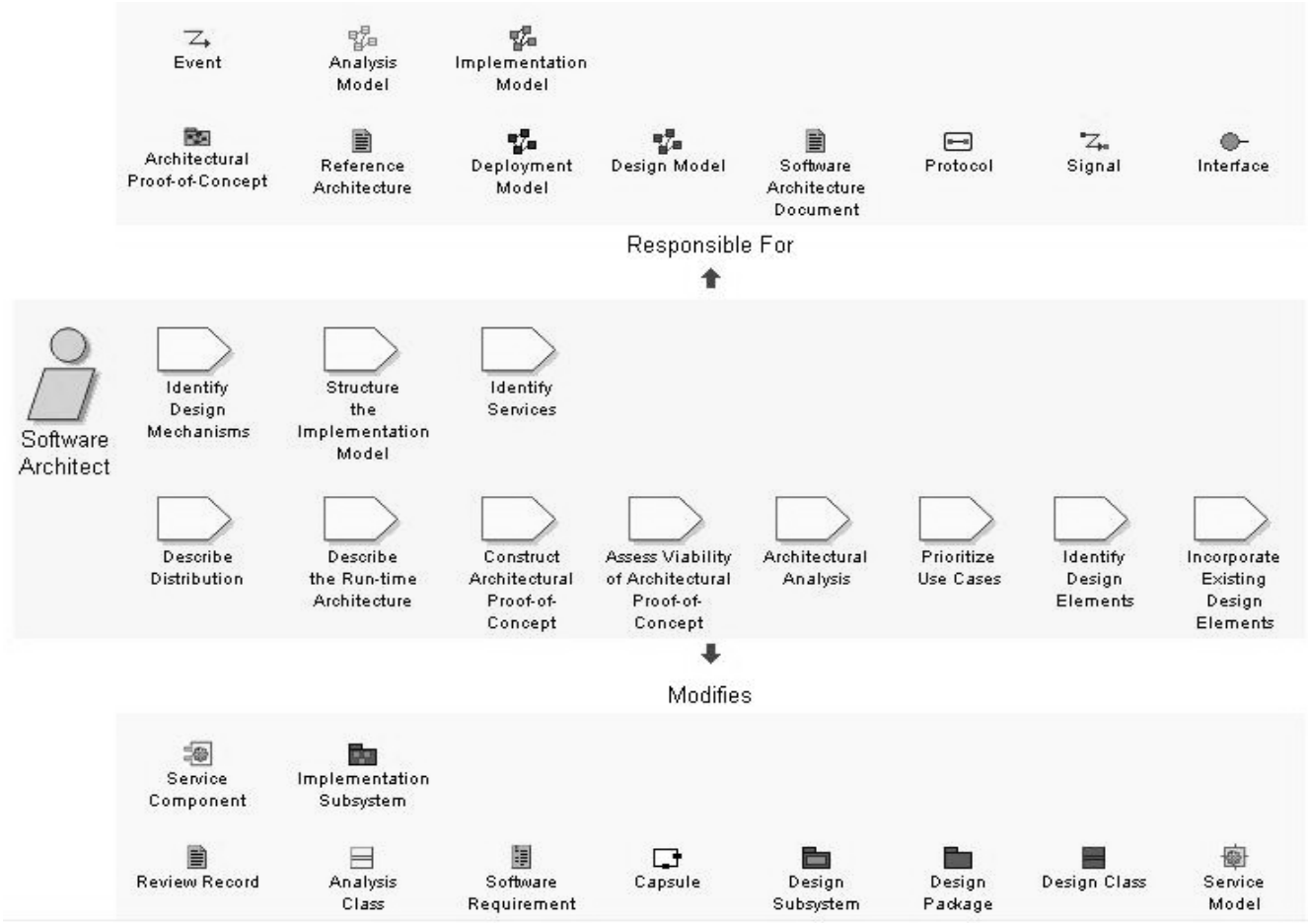


Ilustración 3: Descripción de responsabilidades y actividades del arquitecto de software.

El arquitecto de software tiene la responsabilidad total de conducir las decisiones técnicas principales dentro de la arquitectura del software. Esto incluye identificar y documentar los aspectos arquitectónicamente significativos del sistema, incluyendo requisitos, diseño, implementación, y el despliegue, así como las "vistas" del sistema.

El arquitecto es también responsable de proporcionar la razón fundamental para tomar las siguientes decisiones: balancear las preocupaciones de los stakeholders, conducir los riesgos técnicos, y asegurar que las decisiones han sido comunicadas, validadas y realizadas eficientemente.

2.3.2 Diseñador.

El diseñador, como se muestra en la Ilustración 4 es responsable de diseñar una parte del sistema, dentro de las limitaciones de los requisitos impuestos de la arquitectura y del proceso del desarrollo para el proyecto que se realiza.

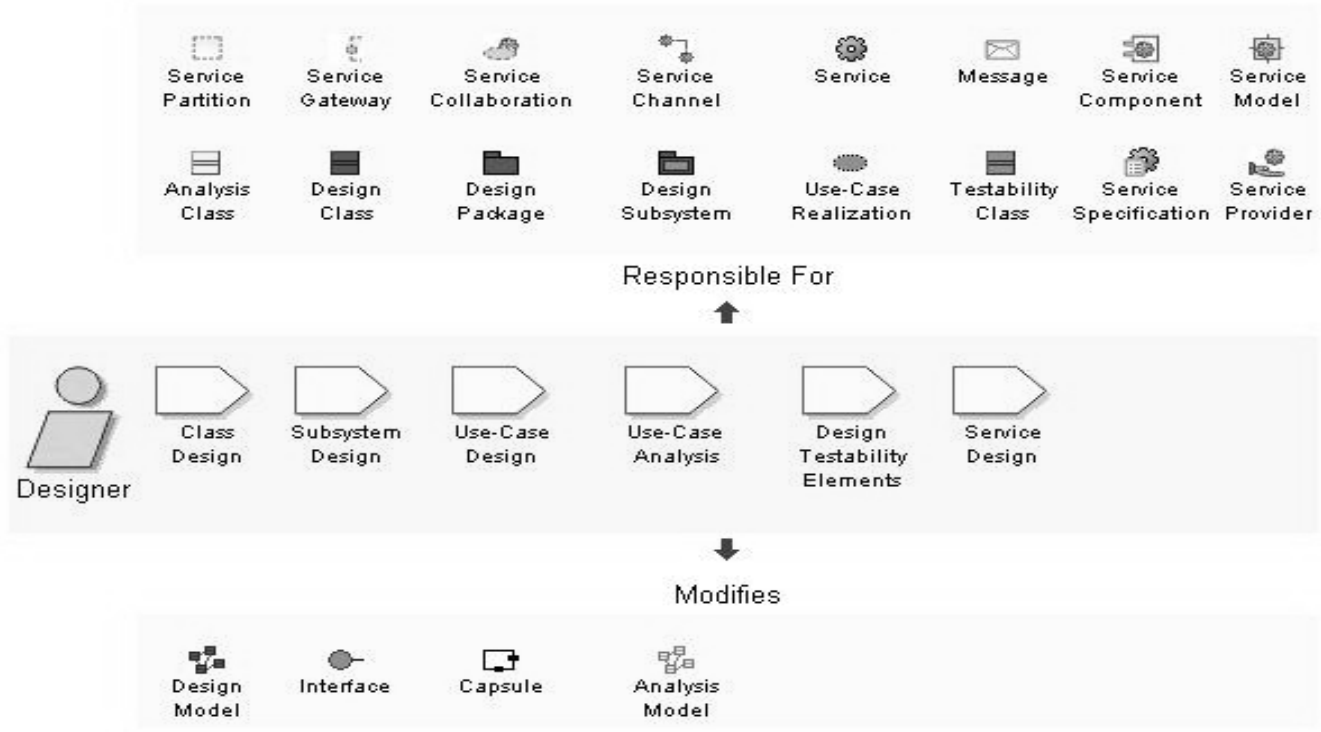


Ilustración 4: Descripción de responsabilidades y actividades del diseñador.

El diseñador identifica y define las responsabilidades, las operaciones, las cualidades, y las relaciones de los elementos del diseño asegurando que el mismo sea consistente con la arquitectura del software, hasta el punto donde pueda asegurarse que la implementación podrá realizarse correctamente.

Habilidades.

El diseñador debe tener conocimientos sólidos acerca de:

- ✓ Requisitos del Sistema.
- ✓ Arquitectura del Sistema.
- ✓ Técnicas de diseño de software, incluyendo técnicas orientadas a objeto, de análisis y de diseño y lenguaje UML.
- ✓ Tecnologías con las cuales el sistema será implementado.

Proyectar las pautas del diseño relacionadas con la implementación, incluyendo el nivel del detalle esperado en el diseño, muy necesario antes de pasar a la implementación del sistema.

2.4 Actividades.

La actividad de identificación de los servicios es ejecutada por el arquitecto de software mientras describe la arquitectura de la solución, la salida es el Modelo de Servicios, el cual contiene un conjunto de servicios identificados y que aún son candidatos para que estos sean refinados por el diseñador como parte de la actividad Diseño de Servicios.

2.4.1 Identificar Servicios.

Propósito.

- ✓ Identificar los elementos de diseño de una solución orientada a servicios en términos de servicios y particiones.
- ✓ Documentar la especificación inicial de servicios.
- ✓ Determinar las dependencias iniciales y la comunicación entre servicios.

Rol: Arquitecto del Software.

Manejo de procesos de negocio.

Esta sección trata sobre cómo el arquitecto del software o el diseñador pueden trabajar en la identificación de servicios. La identificación de servicios es una de las primeras actividades a modelar en una solución orientada a servicios, y por lo tanto los errores cometidos pueden influir más adelante en las actividades del diseño e implementación. En ella el arquitecto

La Ilustración 5 muestra las diferentes propuestas para la identificación del servicio, ellas no son mutuamente excluyentes. La decisión de cual de ellas escoger está muy relacionada con las necesidades propias del proyecto. Cada óvalo coloreado representa una de las técnicas descritas más abajo, curiosamente existen dos formas de identificación basada en la colaboración, una utilizando el Modelo de Procesos del Negocio y otra usando el Modelo de Casos de Uso.

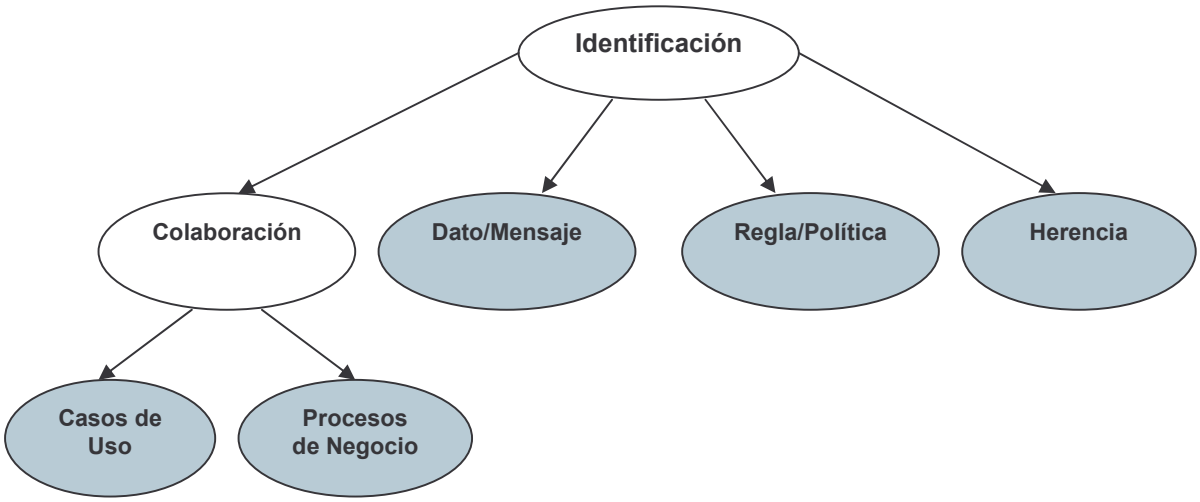


Ilustración 5: Propuestas para la Identificación de Servicios.

Una de las primeras decisiones a tomar, independientemente de las propuestas anteriormente planteadas, está sustentada, en conocer si la identificación está basada puramente en la comprensión de las operaciones que constituyen una agregación en los servicios o de, si un sistema de servicios es identificado a priori y las operaciones están agregadas a los servicios según lo identificado.

Servicio primero: esta técnica es común en el desarrollo orientado a objeto y basado en componentes, donde las clases de objetos o de componentes se identifican primero usando una técnica de análisis para identificar las clases de “cosas” en un cierto negocio o dominio técnico. Como las colaboraciones entre los objetos son analizadas, las operaciones (responsabilidades del objeto) son identificadas y agregadas a las clases. De la misma forma los servicios pueden ser identificados del análisis del dominio.

Operación primero: los servicios no son como las clases y objetos, o los componentes. Los servicios pueden manejar un sistema de recursos, sin embargo la relación servicio/recurso es diferente a la relación

clase/objeto. Estas diferentes técnicas de análisis son necesarias y tienden a favorecer la identificación tardía de los servicios agregando un conjunto de operaciones en una agrupación lógica.

Los ejemplos siguientes demostrarán el uso de la técnica servicio primero de la forma más fácil utilizada por aquellos que están familiarizados con las orientaciones análogas en RUP.

De arriba hacia abajo, dirigido por los procesos del negocio.

En general este acercamiento proporciona una conexión muy fuerte entre los stakeholders/usuarios del negocio y la organización de tecnologías de la información que implementa servicios para que apoyen las tareas identificadas en el modelo de procesos del negocio. En general este modelo está enfocado en tareas realizadas por roles y/o recursos en una organización, usualmente para proveer valores en el formulario de un producto o servicio de una parte externa, como es un cliente o socio.

Este proceso es una selección organizada de algunas tareas, divididas en subprocesos y tiene una organización asociada en recursos y modelos de datos para capturar todos los aspectos de procesos incluyendo no solamente roles, responsabilidades, definiciones de artículos y tareas apartes.

En una solución orientada a servicios el servicio se identifica en un nivel similar de granularidad, es decir, se asume comúnmente que las operaciones en una especificación del servicio corresponderán 1:1 con las tareas atómicas identificadas en un Modelo de Procesos del Negocio. Mientras que esto es un acercamiento atractivo y puede, si está hecho cuidadosamente, alcanzar los resultados correctos, también tiende a conducir a la suposición que tales servicios identificados una vez, los que pueden ser implementados directamente como se describen en el modelo de procesos.

Lo que este enfoque no toma en consideración es que hay requisitos no funcionales que efectúan la clase de servicio a desarrollarse y la manera en que las operaciones se identifican en servicios y así sucesivamente. El nivel del detalle usualmente capturado por tales herramientas no incluye por ejemplo capturar la seguridad, la calidad del servicio o políticas de manejabilidad. Transformar el proceso en un sistema de especificaciones del servicio candidato en un Modelo de Servicios proporciona un punto de partida, a partir del cual se realice el análisis adicional antes de que se desarrolle el Modelo de Diseño que describe la implementación efectiva.

De arriba hacia abajo, dirigido por Casos de Uso.

En el desarrollo de soluciones basadas en componentes y orientadas a objeto existe la tendencia de realizar un conjunto de transformaciones a través de diferentes niveles de abstracción y adicionando niveles de detalle a partir de los CU hasta el diseño del sistema, esto se realiza fundamentalmente cuando se toma como punto de partida los Casos de Uso del Negocio (CUN) a partir de los cuales se desarrolla un Modelo de Diseño.

El artefacto “Casos de Uso del Negocio” describe la relación entre los procesos del negocio y los Casos de Uso del Negocio por lo que se puede definir que el Modelo de Servicios puede ser derivado del Modelo de Casos de Usos del Negocio. En la Ilustración 6 se muestra como se realiza:

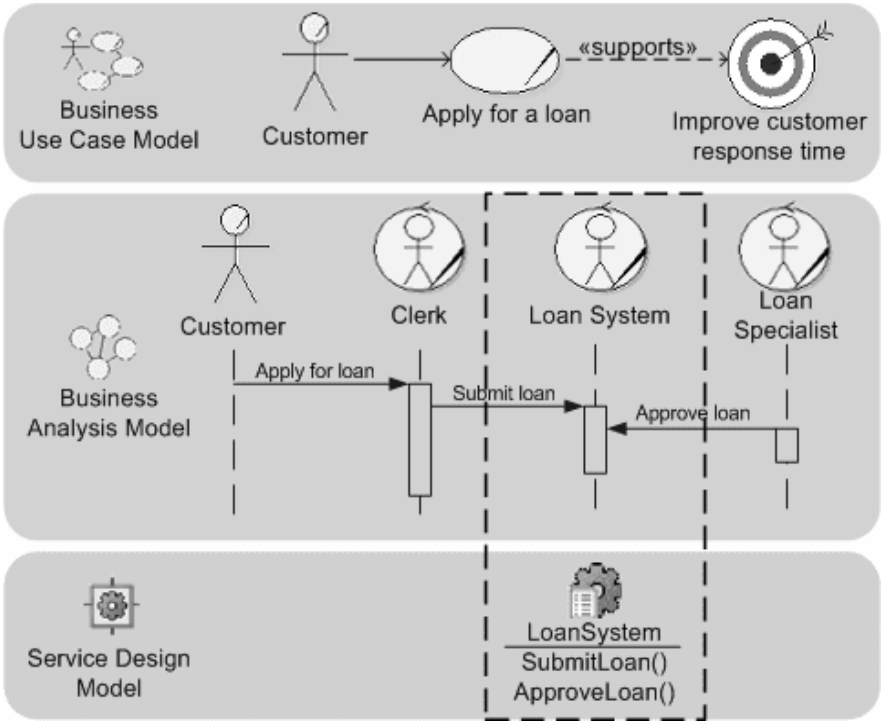


Ilustración 6: Conexión entre el Modelo de Análisis del Negocio y el Modelo de Servicios.

Esta conexión directa entre el Modelo de Análisis del Negocio y el Modelo de Servicios no solo permite que los servicios puedan verse como apoyo a las necesidades del negocio, sino que además aportan

menos transformaciones entre la expresión del negocio que se necesita y la solución, para responder con más eficacia al cambio en los casos de uso del negocio o los modelos del análisis.

Otro aspecto importante es que el Modelo de Caso de Uso del Negocio también incluye las metas de negocio que impulsan el negocio, siendo mucho más fácil identificar realmente la alineación entre los servicios y las metas. Por ejemplo es posible enumerar, para cualquier especificación del servicio todas las metas de negocio a las cuales contribuya; y viceversa para cualquier meta de negocio podemos enumerar los servicios desplegados realmente en nuestra organización IT (**I**nformation **T**echnologies) que contribuye a la meta (siguiendo la conexión del servicio a la especificación).

En el ejemplo de la Ilustración 6 (modelo de arriba), se supone que hay una relación detectable entre las especificaciones del servicio derivadas de los trabajadores del negocio. Tal relación nos permite asegurar que si la definición de los elementos en el Modelo de Análisis del Negocio cambia, y que el diseñador responsable del Modelo de Servicios puede analizar el cambio y su impacto en las especificaciones del servicio y las definiciones del mensaje.

Dirigido por datos.

Es crucial para la realización del análisis y diseño de la solución, entender y administrar los datos o la información del negocio, muchas de estas soluciones incluyen servicios que actúan como administradores de datos por lo que la identificación de servicios tiende a enfocarse más en el Modelo de Datos, Modelo de Dominio o Modelo de Análisis del Negocio. En términos de reingeniería para una solución orientada a servicio, el Modelo de Datos tiene que ser desarrollado a partir de las aplicaciones existentes de forma tal que pueda ser usado para identificar subconjuntos coherentes que serán tratados como servicios autónomos.

Este tema se ocupa más detalladamente adentro de la guía de Encapsulamiento del Servicio de Datos.

Conducido por las reglas del negocio.

Ciertas clases de soluciones tienden a apoyarse fuertemente en el uso de las Reglas de Negocio para extraer la variabilidad de la solución y expresar como esas reglas pueden desarrollarse fuera de la lógica

de la aplicación principal. De un Modelo de Análisis del Negocio incluyendo Entidades del Negocio y Reglas del Negocio es posible definir los servicios que encapsulan las reglas de negocio, exponiéndolas al resto de la lógica de la solución.

El siguiente diagrama (Ilustración 7) contiene una pequeña muestra de un Modelo de Análisis del Negocio mostrando dos reglas del negocio adjuntadas a la entidad del negocio denominada Order. Estas reglas del negocio, como se unen a una entidad de negocio, es más probable que correspondan a las invariantes en la entidad y así serán evaluadas en cualquier cambio en el estado de la entidad. Las reglas se pueden también adjuntar a las acciones, o a los procesos y se evalúan más a menudo como pre-condiciones o post-condiciones para la acción.

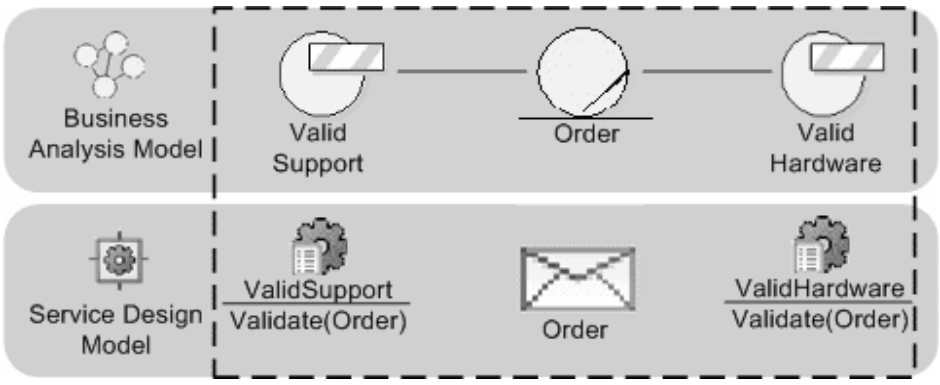


Ilustración 7: Relación entre las especificaciones del servicio y los mensajes.

En el modelo de la ilustración 7, se asume que hay una relación detectable entre las especificaciones del servicio derivadas de las reglas de negocio y los mensajes derivados de la entidad de negocio.

En muchos casos las reglas complejas son agrupadas en un conjunto de reglas, éstos se corresponden mucho más con la granularidad del servicio, permitiendo que por ejemplo, un documento sea pasado al servicio de validación donde el conjunto de reglas son evaluadas y los resultados son retornados. Del ejemplo anterior se puede inferir fácilmente que los servicios de validación incorporan un complejo sistema de reglas para validar la compatibilidad de los temas ordenados, cantidades, etc.

De Abajo hacia Arriba, exponiendo activos existentes.

En la actualidad muchas aplicaciones son construidas sin tener en cuenta las aplicaciones existentes, las cuales pueden brindar funcionalidades que la solución necesita o con las que la solución tiene que interactuar. Por tal motivo es vital que las aplicaciones heredadas (legacy) sean debidamente catalogadas y sus funcionalidades estén correctamente identificadas con el propósito de que las mismas puedan ser reutilizadas como parte de una nueva solución.

En las soluciones orientadas a servicios existen un conjunto de variantes para realizar la integración de los nuevos servicios con las funcionalidades existentes:

1. **Empaquetar funciones existentes como un servicio.** En este caso se deja la función tal como esta pero se utilizan herramientas o middleware para exponer la función existente como un servicio.
2. **Empaquetar y reemplazar funciones existentes con un servicio.** En este caso se empaqueta la función igual que en el caso anterior, pero utilizando la especificación del servicio resultante para desarrollar un nuevo servicio que será más tarde reemplazado por el original y se redireccionará a los clientes a la nueva implementación.
3. **Usar un adaptador más manejable para la invocación del servicio.** En algunos casos no es posible empaquetar la función y exponerla como un servicio, pero puede ser posible empaquetar la función en algo que pueda ser integrado.
4. **Integrar la función dentro del servicio.** En algunos casos es obvio que el nuevo servicio puede acceder a las funciones heredadas mediante el uso de la función como un componente lógico dentro de la implementación del servicio.

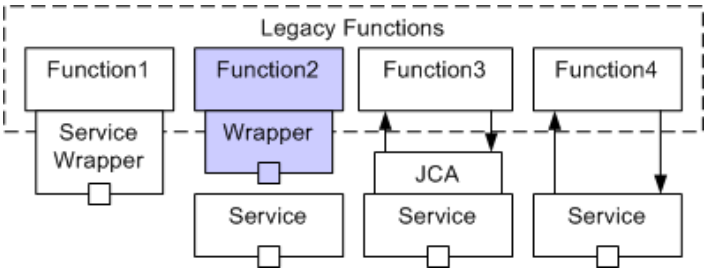


Ilustración 8: Relación de los servicios con funciones de la herencia.

Se considera que tanto la tercera como la cuarta opción brindan una mayor flexibilidad. En ellas se usan las funciones existentes pero no se continúa exponiendo la función tal como es, a los clientes. Por otra parte, la primera y la segunda variantes pueden introducir dificultades en el funcionamiento del protocolo de servicio Web al empaquetar las funciones existentes como servicios. De igual forma la incompatibilidad entre el formato de datos nativo y el XML introduce problemas en el funcionamiento.

Empaquetar activos existentes como un patrón de servicios.

En algunos casos es conveniente desarrollar una partición de servicios a partir de una aplicación heredada, exponiendo un conjunto de funciones de forma individual como servicios. Esta partición solo es accedida por un nivel superior de servicios los cuales la utilizan para una representación más granular de la especificación del negocio a los consumidores. Este encapsulamiento de las funciones heredadas pudiera ser una solución temporal asumida si las características de funcionamiento de la tecnología para realizar el empaquetado de las funciones.

Una forma de ver el empaquetado de una función heredada es una forma muy simple del modelo de proveedor de servicios, con un simple servicio, realizando una simple especificación con una simple operación, en la Ilustración 9 se muestra un ejemplo del patrón “UpdateCustomerAddress”

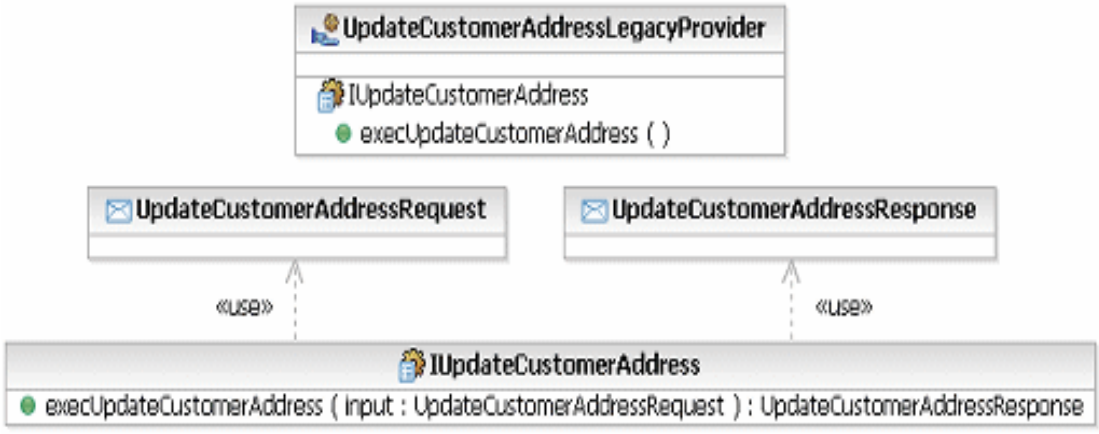


Ilustración 9: Ejemplo del patrón “UpdateCustomerAddress”

2.4.2 Diseño de Servicios.

Propósito.

- ✓ Definir los servicios y la estructura de una solución orientada a servicios.
- ✓ Documentar la especificación del servicio.
- ✓ Determinar las dependencias y la comunicación entre los servicios.

Rol: Diseñador.

Artefactos de Entrada:	**Artefactos Resultantes:**
✓ Modelo de Diseño. ✓ Pautas Específicas del Proyecto. ✓ Servicio. ✓ Modelo de Servicios. ✓ Especificación del Servicio. ✓ Documento de la Arquitectura de Software. ✓ Especificaciones Suplementarias.	✓ Modelo de Diseño. ✓ Servicio. ✓ Modelo de Servicios. ✓ Servicio de Especificación.

Tabla 3: Artefactos de entrada y salida de la actividad Diseño de Servicios.

Carpeta de Servicios de la Empresa.

Una de las ventajas más conocidas al utilizar una Arquitectura Orientada a Servicios es su capacidad para que los servicios representen activos reutilizables de la empresa, o sea, se incorpora el concepto de que una verdadera arquitectura SOA proporciona todas las capacidades de infraestructura y de negocio como servicios y que las aplicaciones desarrolladas se conviertan en capacidades para reutilizar una determinada Carpeta de Servicios.

Cuando se comienza un proyecto, es importante conocer si se están desarrollando los servicios como parte de una Carpeta de Servicios o si se está desarrollando la funcionalidad de aplicaciones que utilizan

estos servicios. Por ejemplo, el desarrollo de un Portal para que los clientes tengan acceso a información de su cuenta bancaria es un proyecto de desarrollo de aplicaciones usando servicios de una Carpeta de Servicios para información del cliente, información de la cuenta, ofertas, etc. En cada caso el uso de la Carpeta de Servicios tiene diversas implicaciones; el diseñador está describiendo su especificación del servicio y publicándola como parte de la lista, esta especificación permite que los desarrolladores de la aplicación entiendan los requisitos de interacción para el servicio. El implementador del servicio utiliza la misma especificación del servicio para desarrollar unas o más implementaciones del servicio, asegurando que la implementación es coherente a la especificación.

Describir los elementos de servicio.

Para modelar sistemas de software es necesario definir una cantidad X de puntos de entrada a cualquier modelo, además de definir una metodología aplicable a nuestro negocio. En la mayoría de los casos, estos puntos de entrada son definidos debido a preocupaciones que surgen sobre la base de la tecnología a utilizar, o del negocio que se modela, lo cual es muy importante para entender que se quiere hacer y asegurar el éxito del proyecto.

Una pequeña cantidad de estas operaciones está dirigida a desarrollar soluciones orientadas a servicios. El siguiente diagrama (Ilustración 10) representa estas preocupaciones primarias como actividades de diseño específicas. Cada uno de estas preocupaciones puede actuar como punto de partida para el diseño del servicio.

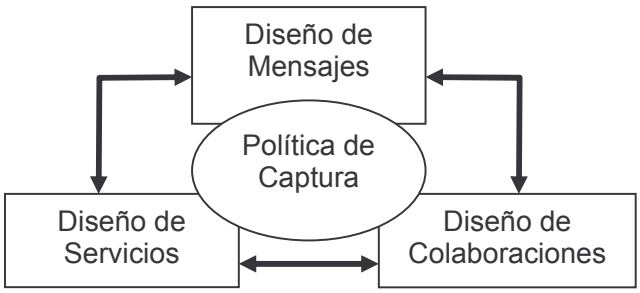


Ilustración 10: Actividades de diseño específicas

Escenarios.

Se identificaron 3 métodos para el desarrollo de los servicios, que no son mutuamente excluyentes, lo cual quiere decir que en un proyecto real se podrá utilizar una combinación de estas técnicas. Los escenarios son:

1. Centrado en el diseño de mensajes.
2. Centrado en el diseño de servicio.
3. Centrado en el diseño de las colaboraciones.

Escenario centrado en el diseño de mensajes.

En este escenario el foco está sobre el dominio del servicio, técnicas como la ingeniería de dominio o el análisis y diseño orientado a objeto brindan una mejor visión en el desarrollo abstracto del modelo de dominio, estas técnicas generalmente nos llevan a la obtención de modelos altamente reutilizables.

Usualmente el Diseño de Servicios es una actividad secundaria a pesar de que puede ser ejecutada en paralelo.

El analista del negocio comienza a documentar los requisitos en el Modelo de Procesos del Negocio, el arquitecto de software toma este modelo y de conjunto con el analista del negocio desarrollan un modelo de procesos más detallado, de igual forma ambos trabajan juntos para obtener una mayor compresión de los requerimientos de datos.

El diseñador de la base de datos trabaja de conjunto con el arquitecto de software en el diseño del esquema para la representación lógica y física de los procesos.

El arquitecto de software desarrolla un Modelo de Servicios candidato, en él, las actividades del proceso se particionan en un conjunto de operaciones de servicio. Este modelo incluye los requerimientos de mensajes para cada servicio y la relación de estos servicios con el modelo de datos realizado.

El arquitecto de software trabaja con el integrador para definir, siempre que existan y sea posible, el mapeo requerido desde el modelo candidato a la implementación actual.

Escenario centrado en el diseño de servicios.

En este escenario el diseñador expone como un servicio o conjunto de servicios las funciones esperadas del negocio o de la aplicación.

En este caso no necesariamente hay que conocer que es lo que el cliente del servicio desea hacer con el mismo, más aún no se conoce que tipo de interacciones el cliente espera del servicio, por consiguiente y contrario al escenario anterior, los mensajes tienden a ser secundarios y son desarrollados en respuesta a los requerimientos de una operación.

El arquitecto de software produce un conjunto de descripciones de CU en las cuales define las capacidades esperadas por los clientes y garantiza que el equipo de clientes esté de acuerdo con ellos. Estos Casos de Uso son divididos en un conjunto de operaciones discretas donde son especificados los requerimientos de datos (mensajes).

Una vez que son conocidos los servicios y el conjunto de operaciones, se comienzan a realizar formalmente las especificaciones, esto incluye los protocolos para cada interfaz y las políticas de información que son requeridas por los consumidores del servicio. Este paso es muy importante por cuanto se les brinda a los consumidores la información que no puede ser entendida mediante la lectura simple de un WSDL (**W**eb **S**ervices **D**escription **L**anguage) desplegado por el negocio cuando los servicios están listos para su uso.

Escenario centrado en el diseño de colaboraciones.

Está enfocado a la colaboración de uno o más servicios. Este es muy parecido a una vista de procesos del servicio y está más relacionado con la disciplina tradicional de modelado de negocio, que con una actividad del desarrollo de software. Aquí los servicios son vistos como una realización de los roles en la colaboración, por eso la especificación del servicio es un conjunto de responsabilidades definidas para el rol a través de una o más colaboraciones.

Usualmente en este escenario, la especificación del servicio puede ser derivada de la colaboración.

Durante este escenario el analista de negocio crea o actualiza un Modelo de Procesos del Negocio. El arquitecto de software realiza una comparación de los requerimientos del nuevo proceso de negocio contra la carpeta de servicios existentes.

El analista del negocio define los principales indicadores de performance como parte del propio proceso de negocio.

El diseñador de base de datos refina la definición de los mensajes del Modelo de Procesos del Negocio, asegurando la estandarización de los esquemas existentes, y después desarrolla un modelo de mensaje que puede ser utilizado para generar el esquema XML requerido por el servicio.

El arquitecto de software actualiza el Modelo de Servicios con los nuevos servicios o adicionándole las nuevas especificaciones a los servicios existentes. El modelo de mensaje desarrollado por el arquitecto de datos es usado o reusado para esta especificación del servicio.

El arquitecto de software y el analista del negocio actualizan el modelo de procesos definido en los inicios para mapear las actividades a los nuevos servicios o actualizarlos, en estos momentos el modelo está completado con relación a la actualización de los servicios, por lo que podrá ser versionado y publicado.

El arquitecto de software y el diseñador desarrollan la especificación detallada del servicio, la cual incluye el protocolo y las políticas de información. Esta especificación detallada actúa como un contrato entre el proveedor y el consumidor del servicio, la cual no podrá ser incumplida por ninguna de las partes.

El integrador actualiza la coreografía del proceso con el nuevo servicio.

Políticas de identificación y captura.

Es importante no sobrecargar el mensaje con información detallada que pueda ser utilizada para descifrar el contenido. Por ejemplo: puede conocerse que cierto mensaje tiene que ser transmitido y tiene que mantenerse ‘privado’ porque contiene información personal de un cliente, la política tiene que ser mantener esta característica privada utilizando técnicas de cifrado de datos como AES 128 bits (**A**dvanced **E**ncryption **S**tandard) para evitar que sea descifrado, utilizando además el cifrado AES 128 sobre un

MODEM de datos canónico XML con certificados X.509 (dicho cifrado no es de dominio público), para mantener la integridad del mensaje.

La Ilustración 11 muestra la asociación de la Política de Identificación y Captura con los elementos del Modelo de Servicios. Puede observarse que la información de la política se puede unir a la información con excepción, de los componentes de la especificación, aunque esta (política de información) es el campo de interés primario.

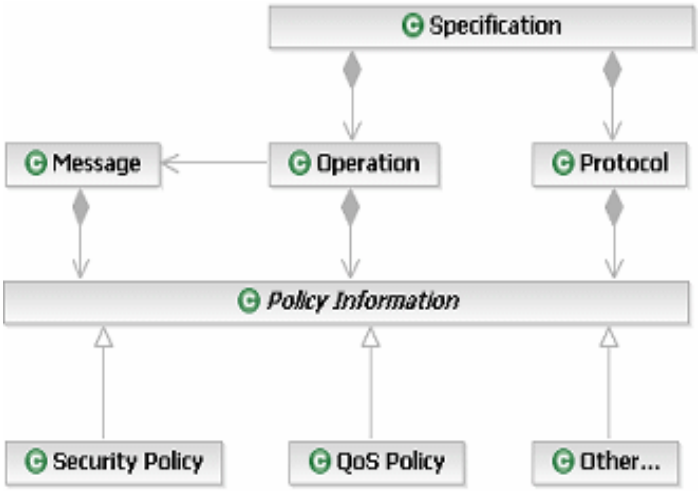


Ilustración 11: Asociación de la política de Identificación y Captura con los elementos del Modelo de Servicios.

Descripción de las dependencias de servicios.

Mientras que en una parte del Modelo de Servicios un número de dependencias se identifican naturalmente, éstas pueden ser tan obvias como la relación entre un servicio y su especificación, o más complejas como por ejemplo, la relación lógica entre dos servicios independientes, porque ambos pueden ser parte de la misma especificación. Estas dependencias pueden ser parte del proceso de decisión que un cliente de servicio realiza al reutilizar un servicio, particularmente si hay múltiples implementaciones.

Tipos de asociaciones y/o dependencias más importantes en el Modelo de Servicios:

- ✓ La relación entre un servicio y los proveedores de servicio que la implementa.
- ✓ La relación entre un servicio y la especificación del servicio que implementa.
- ✓ La relación entre un servicio y cualquier especificación del servicio que requiera.
- ✓ La relación entre un servicio y cualquier canal de servicios que lo conecte con otros servicios (y por lo tanto al servicio en el otro extremo del canal).
- ✓ La relación entre un servicio y cualquier partición de servicios en las cuales el servicio aparezca.

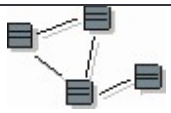
Es por lo tanto importante que todas las especificaciones del servicio sean completas, para una posible reutilización del mismo.

2.5 Artefactos.

Debido a esta nueva actualización para RUP como consecuencia de los cambios presentados en la disciplina Análisis y Diseño se introducen nuevas actividades las cuales llevan asociadas la generación de nuevos artefactos que se incluyen dentro de esta disciplina en particular y en general dentro del proceso de desarrollo de Rational.

2.5.1 Modelo de Diseño.

El Modelo de Diseño es un modelo de objeto que describe la realización de los Casos de Uso del Diseño, y sirve como abstracción del Modelo de Implementación y de su código de fuente. El Modelo de Diseño se utiliza como entrada esencial a las actividades en la disciplina de implementación y prueba.

Estereotipo	
Otras Relaciones	**Contiene** ✓ Protocolo. ✓ Cápsula. ✓ Realización de Casos de Uso. ✓ Señales. ✓ Acontecimientos.

	✓ Subsistema de Diseño ✓ Paquete de Diseño. ✓ Interfaz. ✓ Clases de Diseño. ✓ Clases de Prueba. ✓ Diseño de Prueba. ✓ Componente de Servicios.
Rol	Arquitecto de Software.
Opcionalidad/Ocurrencia	Requerido. Fases de elaboración y de construcción.
Plantillas y Reportes	✓ Reconocer el Modelo del Diseño. ✓ Realización de Casos de Uso. ✓ Subsistema de Paquetes de Diseño. ✓ Reporte de Clases.
Ejemplos	✓ Modelo de Diseño. ✓ CSPS Rose Model.
Representación UML	Modelo estereotipado como <<Modelo de Diseño>>.

Tabla 4: Descripción del Modelo de Diseño.

Propósito.

El Modelo de Diseño es una abstracción de la implementación del sistema. Se utiliza para concebir como documentar el diseño del sistema de software. Es un artefacto que abarca todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

Responsabilidad.

El arquitecto de software es responsable de la integridad del Modelo de Diseño, asegurando que:

- ✓ El Modelo de Diseño se realice correctamente cuando responde en su totalidad a la funcionalidad descrita en el Modelo de Casos de Uso
- ✓ La arquitectura en el Modelo de Diseño satisface su propósito, incluyendo las vistas lógicas, de proceso y de despliegue. Estas vistas se recogen en un artefacto separado, el Documento de la Arquitectura del Software.

2.5.1.1 Componente de Servicios.

El Componente de Servicios es una especialización del subsistema de diseño prevista para utilizarse en la descripción de la realización de una especificación del servicio. Un Componente de Servicios puede proporcionar la realización para uno o más servicios y la realización múltiple de las especificaciones del servicio. El sistema de los elementos del modelo en el interior del componente, representa la realización concreta del contrato estructural, del comportamiento y de la política descrito por la especificación del servicio.

Estereotipo	
Rol	Diseñador.
Opcionalidad/Ocurrencia:	Obligatorio
Representación UML	Componente de UML 2.0, estereotipado como <<Componente de Servicios>>. UML 2.0 proporciona un estereotipo, dentro del perfil “intermedio”, llamado <<servicio>>

Tabla 5: Descripción de Componente de Servicios.

Propósito.

Los componentes de servicios son dominantes al desarrollo de una solución orientada a servicios pues proporcionan la implementación de los servicios identificados dentro del Modelo de Servicios.

El artefacto Componente de Servicios es utilizado por:

- ✓ Implementadores (de los servicios) para describir los elementos del modelo que proporcionan la implementación del comportamiento del servicio.

El Componente de Servicios proporcionará un encapsulamiento completo de su comportamiento y expondrá solamente las capacidades definidas por la especificación del servicio. Esta especificación también incluye las especificaciones del comportamiento.

Propiedades.

Nombre Propiedad	Una breve descripción	Representación UML
Nombre	Nombre de la clase.	La cualidad "nombre" en el elemento modelo.
Breve descripción	Breve descripción del rol y del propósito de la clase.	Valor marcado con etiqueta, del tipo "texto corto".
Servicios Realizados	El sistema de especificaciones de servicio observadas por este componente de servicios.	Todas las relaciones de la realización del componente de servicios a las especificaciones del servicio.
Requerimientos Especiales	Necesita una descripción textual que recoja todos los requisitos, tales como requisitos no funcionales, teniendo en cuenta las clases en el Modelo de Diseño y su importancia a la hora de la implementación.	Valor marcado con etiqueta, del tipo "texto corto".

Tabla 6: Propiedades del Componente de Servicios.

Responsabilidad.

El diseñador es el responsable de la integridad del componente, asegurando que:

- ✓ El componente encapsula su contenido y sólo expone el comportamiento contenido a través de interfaces realizadas del Modelo de Servicios.
- ✓ Las operaciones de las interfaces que el componente realiza se distribuyen a las clases o a los subsistemas contenidos.
- ✓ El componente expone correctamente sus interfaces en la implementación según las especificaciones estructurales y de comportamiento definida en el Modelo de Servicios.
- ✓ El componente implementa adecuadamente las políticas de chequeo y las restricciones identificadas en la especificación del servicio.

Adaptación.

Los componentes de servicios representan la realización de los servicios identificados en el Modelo de Servicios y descritos por la especificación. Un componente de servicios se puede descomponer en componentes o clases dentro del diseño de su implementación. A continuación se identifican estereotipos adicionales, además del uso del componente estándar, clases y elementos del Modelo de Diseño de RUP para patrones de componentes de servicios:

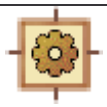
Icono	Nombre	Representación UML	Descripción
	**Fachada**	**Estereotipo en clase o componente.**	Denota el componente que actúa como fachada para la implementación del servicio. En general existe un componente de fachada para cada especificación observada del servicio.
	**Mediador**	**Estereotipo en clase o componente.**	Es utilizado en situaciones donde puede haber una o más implementaciones para una operación dada del servicio, donde la fachada llama al mediador para identificar y para llamar el

			componente correcto de implementación.
	**Acceso a datos**	**Estereotipo en clase o componente.**	Denota un componente de acceso a datos, el mismo es responsable del acceso y de la gerencia de los datos persistentes para la implementación del servicio.

Tabla 7: Estereotipos adicionales para patrones de componentes de servicios.

2.5.2 Modelo de Servicios.

El modelo representa una abstracción arquitectónica para las soluciones orientadas a servicios permitiéndoles a los arquitectos y diseñadores desarrollar los modelos donde la noción de un servicio y sus especificidades son elementos de primera clase. Este modelo no es una representación directa de ninguna tecnología de implementación tal como WSDL.

Estereotipo:	
Otras relaciones:	Contiene. ✓ Mensaje ✓ Servicio ✓ Canal de Servicios. ✓ Colaboración de Servicio. ✓ Entrada de Servicio. ✓ Partición de Servicios. ✓ Proveedor de Servicios. ✓ Especificación del Servicio.
Rol	Diseñador

Plantillas e Informes	✓ Reporte: Carpeta de Servicios. ✓ Reporte: Dependencias de Servicio. ✓ Reporte: Meta de trazabilidad del Servicio
Representación UML.	Modelo estereotipado como <<Modelo de Servicios>>.

Tabla 8: Descripción de Modelo de Servicios.

Propósito.

Se utiliza para documentar el diseño de los servicios del software. Es un artefacto comprensivo, que abarca todos los servicios, proveedores, especificaciones, particiones, mensajes, colaboraciones, y las relaciones entre ellos. Constituye una abstracción de los servicios implementados dentro de una empresa y de apoyar el desarrollo de una o más soluciones orientadas a servicios.

Responsabilidad.

El diseñador es responsable de la integridad del Modelo de Servicios, asegurando que:

- ✓ El Modelo de Servicios se realice correctamente cuando responde en su totalidad a la funcionalidad descrita en el Modelo de Casos de Uso.
- ✓ El refinamiento gradual de la especificación abstracta con la especificación detallada se realiza para que los implementadores desarrollen correctamente el servicio.

El arquitecto de software no es responsable de los paquetes, de las clases, de las relaciones, de las realizaciones de los CU del diseño y de los diagramas: éstos están bajo la responsabilidad de los diseñadores correspondientes y del diseñador de casos de uso.

2.5.2.1 Servicio.

Un servicio es un elemento del Modelo de Servicios el cual es la base de una Arquitectura Orientada a Servicios. Es proporcionado por un proveedor de servicios y posee una especificación del servicio.

Estereotipo:	
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador.
Opcionalidad/Ocurrencia	Obligatorio.
Representación UML	Puerto (UML 2.0), estereotipado como <<Servicio>>. Un servicio realizará una interfaz estereotipada como << Especificación del Servicio >>.

Tabla 9: Descripción del artefacto Servicio.

Propósito.

El artefacto Servicio es utilizado por:

- ✓ **Implementadores:** para una comprensión de los roles que desempeña el servicio y de cómo la especificación del servicio es utilizada por el servicio.
- ✓ **Diseñadores** de otros servicios para la comprensión de las colaboraciones en las cuales participan los servicios.
- ✓ **Diseñadores de Casos de Uso:** en realizaciones de Casos de Uso.
- ✓ **Los que diseñan la versión siguiente del sistema** para entender la funcionalidad en el Modelo de Servicios.
- ✓ **Los que prueban las clases** para planear actividades de prueba.

2.5.2.2 Proveedor de Servicios.

Un Proveedor de Servicios agrupa un sistema relacionado de servicios.

Estereotipo	
Otras Relaciones	Parte de Modelo de Servicios
Rol	Diseñador
Opcionalidad /Ocurrencia:	Obligatorio
Representación UML	Clase o Componente, estereotipado como <<Proveedor de Servicios>>. El Proveedor de Servicios no tendrá ningunas operaciones, atributos o el comportamiento especificado fuera de la implementación por los servicios. Cualquier Puerto en el Abastecedor de Servicio será estereotipado <<Servicio>>

Tabla 10: Descripción del artefacto Proveedor de Servicios.

Propósito.

El artefacto Proveedor de Servicios es utilizado por:

- ✓ **Implementadores** para una comprensión de la agregación de servicios y por lo tanto del impacto posible en las opciones del despliegue.
- ✓ **Diseñadores** de servicios para entender las restricciones de agrupar servicios.
- ✓ **Los que diseñan la versión siguiente del sistema** para entender la funcionalidad en el Modelo de Servicios y específicamente las restricciones en servicios móviles de los proveedores.
- ✓ **Los que prueban las clases** para planear actividades de prueba.

Responsabilidad.

El diseñador es el responsable de la integridad del modelo, asegurando que:

- ✓ Los proveedores de servicios puedan apoyar solamente un sistema fijo de opciones obligatorias identificadas para asegurarse que los clientes no seleccionen un atascamiento para un canal de servicios que no pueda ser apoyado.

Se recomienda que las decisiones sobre la asignación de servicios a los proveedores de servicio se centren en la necesidad de la co-localización de servicios. Los servicios que son lógicamente relacionados se pueden identificar usando particiones del servicio.

2.5.2.3 Especificación del Servicio.

Una Especificación del Servicio es un elemento del Modelo de Servicios que proporciona una descripción estructural y de comportamiento para un caso específico del servicio. Una Especificación del Servicio también puede identificar un sistema de políticas que gobiernan el acceso al servicio o el uso del mismo.

Esta especificación actúa como un contrato entre el cliente del servicio y el implementador del servicio. El cliente entiende cómo obrar recíprocamente con un servicio y el implementador entiende el comportamiento esperado de la implementación del servicio.

Estereotipo	
Otras Relaciones	Parte del Modelo de Servicios
Rol	Diseñador
Opcionalidad/Ocurrencia	Obligatorio
Representación UML	Interfaz. Estereotipado como <<Especificación del Servicio>>. Una especificación debe también proporcionar cualquier estado o colaboración que valide su especificación del comportamiento.

Tabla 11: Descripción del artefacto Especificación del Servicio.

Propósito.

El artefacto Especificación del Servicio es utilizado por:

- ✓ **Implementadores** (de los servicios) para una comprensión de la interfaz del servicio que proporcionan, pero también para el comportamiento que sus clientes esperan.

- ✓ **Implementadores** (de los clientes del servicio) para una comprensión de la interfaz del servicio que proporcionan.
- ✓ **Diseñadores de servicios** para entender la relación entre las especificaciones y la relación entre los servicios y las especificaciones que ellos (los servicios) implementan.
- ✓ **Los que diseñan** la versión siguiente del sistema para entender la funcionalidad en el Modelo de Servicios.
- ✓ **Los que prueban las clases** para planear actividades de prueba.

Responsabilidad.

El diseñador es el mayor responsable de la integridad del modelo, asegurando:

- ✓ Que todos los parámetros de la operación en especificaciones del servicio están especificados en términos de mensajes.

Las Especificación del Servicio se pueden reutilizar y componer.

2.5.2.4 Canal de Servicios.

Un canal de servicios es un elemento del Modelo de Servicios que representa una conexión entre dos servicios, o entre un cliente y un servicio. Es importante entender que el canal no representa ninguna interacción en particular. Es utilizado para modelar los mensajes que son enviados entre los servicios que participan en la conexión.

	
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador
Opcionalidad/Ocurrencia:	Obligatorio
Representación UML	Conector (UML 2.0), estereotipado como << Canal de Servicios >>.

Tabla 12: Descripción del artefacto Canal de Servicios.

Propósito.

El Canal de Servicios proporciona la conexión entre dos servicios o entre un servicio y un cliente. También es utilizado para modelar colaboraciones entre los servicios y específicamente los mensajes que son enviados entre los servicios que tienen lugar en la conexión.

2.5.2.5 Mensaje.

Un mensaje es un contenedor de información que identifica un subconjunto del modelo de información o del modelo de dominio, el cual es invocado dentro o fuera del servicio. Un mensaje no debe seguir ningún comportamiento definido.

Estereotipo:	
Otras Relaciones:	Parte del Modelo de Servicios.
Rol:	Diseñador
Opcionalidad/Ocurrencia:	Opcional, se utiliza el elemento modelo donde las estructuras mensaje-específico y mensaje-vacío están dentro del propio modelo de UML.
Representación UML	Un mensaje no tendrá operaciones o especificaciones de comportamiento definidas.

Tabla 13: Descripción del artefacto Mensaje.

Propósito.

El artefacto Mensaje es utilizado por:

- ✓ **Implementadores:** para el desarrollo del esquema que describe las estructuras de la implementación específica del mensaje.

- ✓ **Diseñadores:** para compartir y reutilizar otros servicios y para la comprensión de la información entre especificaciones del servicio.
- ✓ **Arquitectos de datos:** en la comprensión de la relación entre el modelo de implementación neutral del dominio y las representaciones de implementación específicas, tales como el esquema de la base de datos o el mensaje.

El mensaje es opcional y usado para quitar ambigüedades de las estructuras del mensaje de otros elementos que representan el mismo modelo de dominio.

Donde no hay modelo separado del dominio o donde los modelos separados son utilizados para la definición del dominio y del mensaje, el uso del estereotipo explícito del mensaje es innecesario.

2.5.2.6 Colaboración de Servicio.

La Colaboración de Servicio es una representación de un sistema de comunicación entre dos o más servicios encapsulados usualmente como un nuevo servicio. De esta manera el modelo puede representar los servicios cuya implementación es simplemente la colaboración de un sistema de servicios existentes.

Estereotipo:	
Otras Relaciones	Parte del Modelo de Servicios
Rol	Diseñador
Opcionalidad/Ocurrencia:	Opcional
Representación UML	Colaboración, estereotipada como <<Colaboración de Servicio>>. Los participantes en la colaboración pueden solamente ser instancias de los Proveedores de Servicios.

Tabla 14: Descripción del artefacto Colaboración de Servicio.

Propósito.

El artefacto Colaboración de Servicio es utilizado por:

- ✓ **Implementadores** para una comprensión de las colaboraciones y de cómo los servicios deben ser compuestos.
- ✓ **Diseñadores** de servicios en entender el contexto de la colaboración en la cual los servicios serán utilizados y reutilizados.
- ✓ **Los que diseñan la versión siguiente del sistema** para entender la funcionalidad en el Modelo de Servicios.
- ✓ **Los que prueban las clases** para planear actividades de prueba.

Responsabilidad.

El diseñador es sobre todo responsable de la integridad del modelo, asegurando que:

- ✓ Los servicios que participan en una colaboración puedan lógicamente identificar y participar en la colaboración.

2.5.2.7 Partición de Servicios.

Una Partición de Servicios es un elemento del Modelo de Servicios que proporciona un agrupamiento lógico para los proveedores de servicio. Lógico en el sentido que la estructura de partición puede reflejar una estructura del sistema que afecta la manera en que se despliega el sistema físico o puede representar una estructura que no tenga ningún impacto en el despliegue.

Estereotipo	
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador.
Opcionalidad/Ocurrencia:	Opcional
Representación UML	Clase, Componente o Nodo, estereotipado como <<Partición de Servicios>>. La Partición de Servicios no tendrá ningunas operaciones o cualidades, no tendrá ningún comportamiento especificado y no realizará ningún

	interfaz. Cualquier puerto en la partición de servicios será estereotipado como <<Partición de Servicios>> y cualquier estructura compuesta especificará solamente las piezas que son proveedores de servicios.

Tabla 15: Descripción del artefacto Partición de Servicios.

Propósito.

El artefacto Partición de Servicios es utilizado por:

- ✓ **Arquitectos del software** para una lógica partición de una solución y para la definición de interfaces entre tales particiones.
- ✓ **Diseñadores** de servicios en entender la organización lógica de la solución.
- ✓ **Los que diseñan la versión siguiente del sistema** para entender la funcionalidad en el Modelo de Servicios y específicamente en la arquitectura.
- ✓ **Los que prueban las clases** para planear actividades de prueba.

2.5.2.8 Entrada de Servicio.

Una Entrada de Servicio puede verse como un elemento del Modelo de Servicios a menos que ésta no represente un límite en términos de la implementación de la especificación del servicio.

Estereotipo	
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador.
Opcionalidad/Ocurrencia:	Opcional
Representación UML	Puerto, estereotipado como <<Entrada de Servicio>>. Es un tipo de especificación del servicio y será utilizado solamente en particiones del servicio.

Tabla 16: Descripción del artefacto Entrada de Servicio.

Propósito.

El artefacto Entrada de Servicio es utilizado por:

- ✓ **Arquitectos de software** para una comprensión de la comunicación entre las particiones.
- ✓ **Implementadores** para una comprensión de los requisitos de mediación entre las particiones.
- ✓ **Los que diseñen la versión siguiente del sistema** para comprender la composición de particiones y servicios en el Modelo de Servicios.

El diseñador tiene su mayor responsabilidad en la integridad del modelo, asegurando que:

- ✓ Los proveedores de servicios puedan asegurar que los clientes no seleccionen un atascamiento para un canal de servicios al que no se le pueda brindar soporte.

Se recomienda que las decisiones sobre la asignación de servicios a los proveedores de servicios se centren en la necesidad de la co-localización. Los servicios que son lógicamente relacionados se pueden identificar usando particiones de servicios.

2.6 Grafo que describe el proceso del plug-in.

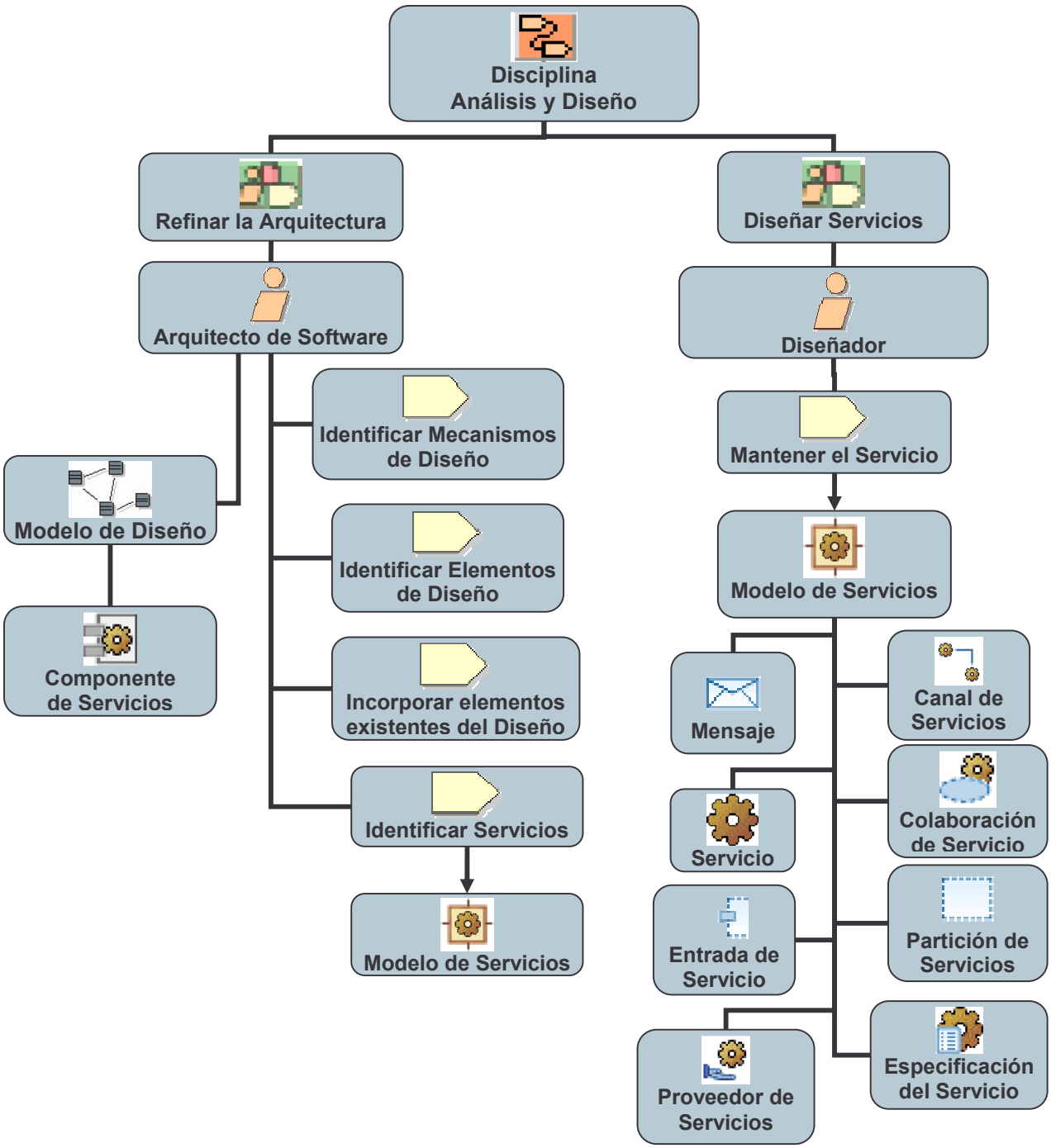

Disciplina
Análisis y Diseño
Refinar la Arquitectura
Arquitecto de Software
Modelo de Diseño
Componente
de Servicios
Identificar Mecanismos
de Diseño
Identificar Elementos
de Diseño
Incorporar elementos
existentes del Diseño
Identificar Servicios
Modelo de Servicios
Diseñar Servicios
Diseñador
Mantener el Servicio
Modelo de Servicios
Mensaje
Servicio
Entrada de
Servicio
Proveedor de
Servicios
Canal de
Servicios
Colaboración
de Servicio
Partición de
Servicios
Especificación
del Servicio

Ilustración 12: Proceso del Plug-in.

Conclusiones.

El plug-in SOA de IBM, constituye una actualización para el proceso unificado del software. Cuenta con una serie de nuevos conceptos y la introducción de nuevas actividades, de donde son generados los nuevos artefactos. Estos son adicionados al negocio anterior, como parte de un Modelo de Servicios introducido en la disciplina de Análisis y Diseño.

El propósito de este capítulo, fue describir detalladamente la actualización del desarrollo con que cuenta RUP a partir de la definición de estos nuevos conceptos y artefactos. Esta metodología es la más eficiente y utilizada a nivel mundial, por su capacidad de ser adaptable a grandes y pequeños proyectos, orientada a objetos y que reúne las buenas prácticas de cada una de las características fundamentales de las metodologías anteriores.

CAPÍTULO 3 CONFIGURACIÓN DEL PROCESO PARA EL PROYECTO ATENCIÓN PRIMARIA DE SALUD

La idea principal que se quiere alcanzar con la introducción de esta nueva actualización para la Suite de Rational es brindar una mayor flexibilidad en la utilización y reutilización de los servicios, aprovechando las libertades de comunicación entre aplicaciones, que brinda la Arquitectura Orientada a Servicios, así como la posibilidad de exponer los procesos del negocio como servicios.

Los principales cambios para esta nueva perspectiva del proceso unificado del software para aplicaciones SOA ocurren durante la disciplina de Análisis y Diseño con la creación de un Modelo de Servicios que abarca todos los servicios y las relaciones entre ellos. También se han introducido nuevos conceptos dentro de las demás fases y disciplinas de RUP para dar un seguimiento adecuado hasta llegar a la creación del Modelo de Servicios, el cual constituye la base de la Arquitectura Orientada a Servicios. Durante este capítulo se describe y configura un proceso de desarrollo sobre la base de soluciones orientadas a servicios para las aplicaciones del proyecto de Atención Primaria de Salud.

3.1 Disciplina Administración de Proyecto. Iteración Cero.

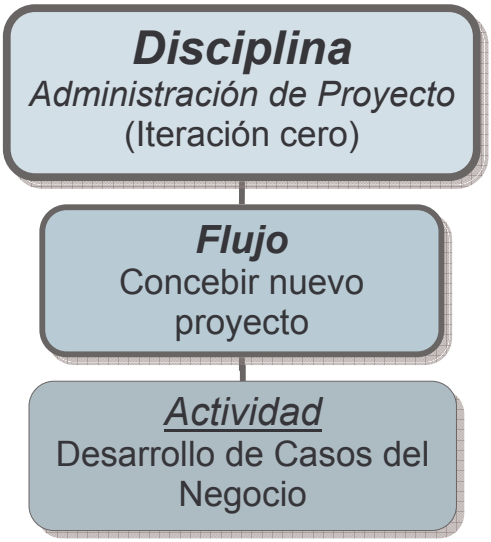


Ilustración 13: Disciplina Administración de Proyecto.

El objetivo principal de esta disciplina es brindar una idea inicial del proyecto de forma tal que se pueda tomar la decisión de construir o no el mismo.

3.1.1 Flujo: Concebir Nuevo Proyecto.

El propósito de este detalle del flujo de trabajo es llevar un proyecto de una idea inicial a un punto en el cual se pueda tomar una decisión razonable para continuar o abandonar el mismo.

3.1.1.1 Actividad: Desarrollo de Casos del Negocio.

Durante el desarrollo de esta actividad se tiene en cuenta la justificación económica para el producto desde el punto de vista de la reutilización de los servicios tanto internos como externos. Esto disminuye los costos a la hora de implementar las aplicaciones, pero se necesitará un mayor esfuerzo para identificar los servicios y validar que los que son seleccionados cumplen con los requisitos.

3.2 Disciplina Modelo del Negocio.

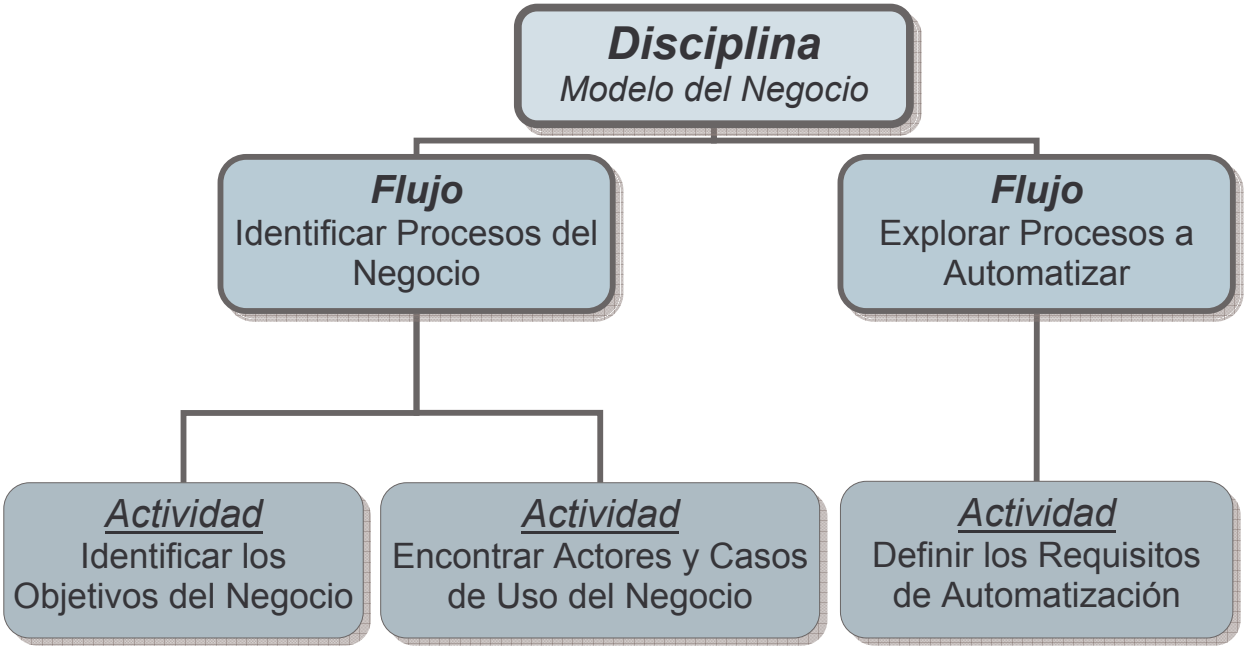


Ilustración 14: Disciplina Modelo del Negocio.

Esta disciplina mantiene sus propósitos iniciales al igual que antes de la introducción de esta actualización para SOA:

- ✓ Entender los problemas actuales de la organización e identificar mejoras potenciales.
- ✓ Evaluar el impacto del cambio organizacional.
- ✓ Asegurar que los clientes, usuarios finales y desarrolladores posean una comprensión común de la organización.
- ✓ Derivar los requisitos del sistema de software necesitados para apoyar la organización
- ✓ Entender como es desplegado un adecuado sistema de software dentro de la organización.

3.2.1 Flujo: Identificar Procesos del Negocio.

El propósito de este detalle del flujo de trabajo es identificar y priorizar los procesos del negocio que necesitan ser descritos detalladamente para una mejor comprensión de los mismos.

3.2.1.1 Actividad: Identificar los Objetivos del Negocio.

La actividad Identificar Objetivos del Negocio tiene como propósito:

- ✓ Identificar los objetivos con las cuales el negocio puede ser planeado y ser manejado.
- ✓ Asegurar la alineación entre los objetivos estratégicos a largo plazo y los objetivos operacionales a corto plazo.
- ✓ Traducir la estrategia de negocio en la acción.
- ✓ Proporcionar una base para cuantificar y mejorar las actividades del negocio.

Las principales fuerzas que afectan el negocio así como las estrategias del mismo son descritas en el artefacto Objetivos del Negocio. Es fundamental que quede descrita la relación de los objetivos con los servicios que están siendo desarrollados para soportar el negocio.

3.2.1.2 Actividad: Encontrar Actores y Casos de Uso del Negocio.

La actividad Encontrar Actores y Casos de Uso del Negocio tiene como propósito:

- ✓ Definir los límites del negocio que se modelará.
- ✓ Definir quién y qué interactuará recíprocamente con el negocio.
- ✓ Esbozar los procesos en el negocio.
- ✓ Crear los diagramas de Modelos de Casos de Uso.

Esta actividad es fundamental en la identificación de los procesos; es precisamente durante la realización del artefacto Casos de Uso del Negocio que podrán ser identificados los servicios candidatos, de igual forma podrán ser identificados a partir del artefacto Reglas del Negocio.

Esta necesidad de lograr un detallado modelamiento del negocio enfocado en los servicios es debido a la capacidad de la Arquitectura Orientada a Servicios de exponer los procesos de negocio como servicios.

3.2.2 Flujo: Explorar Procesos a Automatizar.

El propósito de este detalle del flujo de trabajo es explorar qué porciones de los procesos del negocio pueden y deben ser automatizadas.

3.2.2.1 Actividad: Definir los Requisitos de Automatización.

Esta actividad tiene como objetivo principal obtener a partir del Modelo de Negocio los requerimientos del sistema, atendiendo específicamente al nivel de automatización requerido que de soporte a las actividades del negocio.

Dado que SOA expone los procesos de negocio como servicios es importante que la arquitectura del negocio en la Arquitectura Orientada a Servicios quede reflejada.

3.3 Disciplina Requisitos.

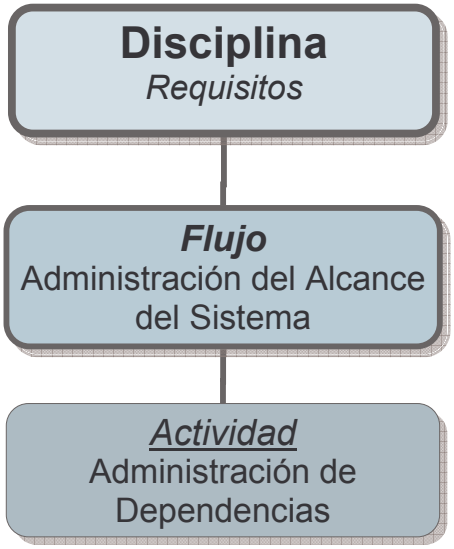


Ilustración 15: Disciplina Requisitos.

La disciplina Requisitos tiene como propósito:

- ✓ Establecer y mantener el acuerdo con los clientes y otros stakeholders en lo que debe hacer el sistema.
- ✓ Definir los límites (delimitar) del sistema.
- ✓ Proporcionar una base para planear el contenido técnico de las iteraciones.
- ✓ Proporcionar una base para el costo de estimación y tiempo de desarrollo del sistema.
- ✓ Definir una interfaz de usuario para el sistema, centrándose en las necesidades y objetivos de los usuarios.

3.3.1 Flujo: Administración del Alcance del Sistema.

El propósito de este detalle del flujo de trabajo es delimitar el alcance del sistema y que el mismo sea desarrollado tan explícito como sea posible.

3.3.1.1 Actividad: Administración de Dependencias.

Esta actividad tiene como propósito:

- ✓ Utilizar las cualidades y la trazabilidad de los requisitos del proyecto para asistir en la administración del alcance del proyecto y la administración de los requisitos cambiados.

Los requisitos de la solución deben ser manejados teniendo en cuenta la autonomía de los servicios. Los servicios con pocas dependencias (tal como ocurre con la Arquitectura Basada en Componentes) son más autónomos y por tanto más reusables. Es importante que los servicios identificados para un sistema particular estén relacionados en la Carpeta de Servicios existente en la empresa.

Carpeta de Servicios

En el caso de las aplicaciones del proyecto APS atendido por la empresa de producción de software SOFTEL existe esta carpeta a pesar de que no se maneja con ese nombre. Dicho ejemplo es el Portal de Arquitectura de SOFTEL en el cual están publicados un conjunto de servicios con su WSDL, como es el caso del RCIE, RPSAP, entre otros, los cuales están disponibles para ser reutilizados por otros proyectos u organizaciones que estén interesadas en su funcionalidad.

3.4 Disciplina Análisis y Diseño.

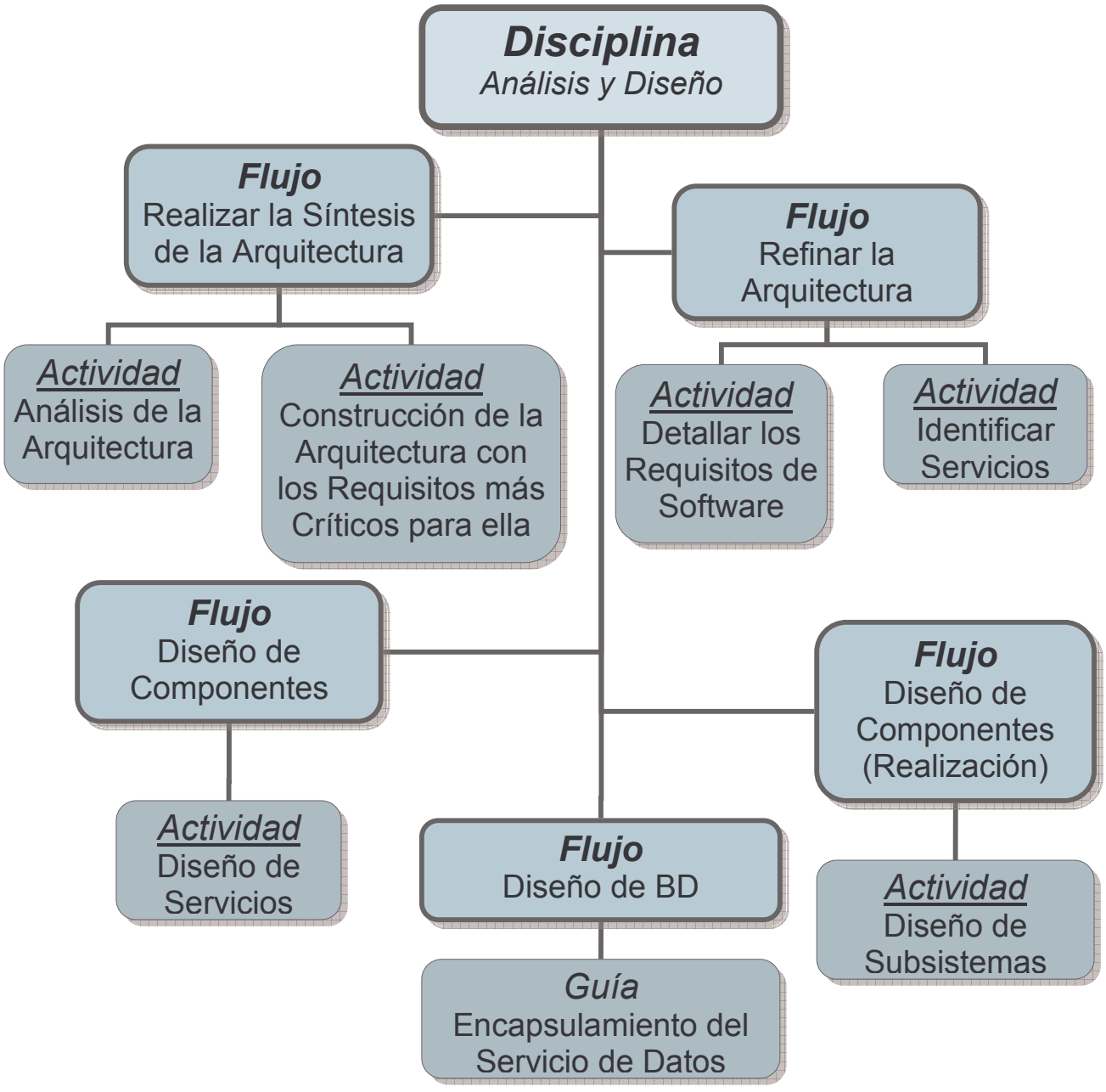


Ilustración 16: Disciplina Análisis y Diseño.

El objetivo de esta disciplina es traducir los requisitos a una especificación que describe cómo implementar el sistema.

3.4.1 Flujo: Realizar la Síntesis de la Arquitectura.

El propósito de este detalle del flujo de trabajo es construir y determinar la construcción de la arquitectura con los requisitos más críticos para ella y demostrar que el sistema, según lo previsto, es factible.

3.4.1.1 Actividad: Análisis de la Arquitectura.

El propósito de esta actividad es:

- ✓ Definir una arquitectura candidata para el sistema, basado en la experiencia ganada de sistemas similares o en dominios de problemas similares.
- ✓ Definir los patrones arquitectónicos, los mecanismos dominantes y convenciones a modelar para el sistema.

La descripción existente anteriormente en el proceso de Rational es cambiada un tanto por la introducción del enfoque de servicio. En primer lugar está disponible la Carpeta de Servicios (Portal de Arquitectura de SOFTEL) para obtener de ella las capacidades que serán reutilizadas de los servicios que contiene. En segundo lugar hay que tener en cuenta la capa intermedia a utilizar para la implementación y despliegue del servicio.

3.4.1.2 Actividad: Construcción de la Arquitectura con los Requisitos más Críticos para ella.

Su propósito general consiste en:

- ✓ Sintetizar por lo menos una solución (que puede simplemente ser conceptual) que reúna los requisitos arquitectónicos más críticos.

Esta actividad puede ser valiosa en la comparación de algunas características del diseño de la distribución y los mensajes que son particulares a las soluciones orientadas a servicios.

3.4.2 Flujo: Refinar la Arquitectura.

El propósito de este detalle del flujo de trabajo es completar la arquitectura para una iteración

3.4.2.1 Actividad: Detallar Requisitos de Software.

El propósito principal de esta actividad es:

- ✓ Colectar, detallar y organizar el conjunto (paquete) de artefactos que describe completamente los requisitos del software del sistema o subsistema.

Dicha actividad se centra en los requisitos técnicos y no funcionales y las restricciones impuestas en los servicios construidos o comprados. Los requisitos no funcionales a considerar durante el desarrollo de esta actividad son los de funcionamiento, de seguridad, infraestructura, dependencias de servicios y las licencias de uso para la ejecución de determinado servicio

En el caso de este último requisito es importante definir los términos de contrato para cada uno de los servicios disponibles para otros proyectos u organizaciones. En simples palabras se debe documentar en un acuerdo o contrato las condiciones de usabilidad de los servicios que se brindan en el Portal de Arquitectura de SOFTEL, asegurando por demás un mantenimiento y soporte adecuado de los mismos.

3.4.2.2 Actividad: Identificar Servicios.

El propósito de esta actividad se centra en:

- ✓ Identificar los elementos de diseño de una solución orientada a servicios en términos de servicios y particiones.
- ✓ Documentar la especificación inicial del servicio.
- ✓ Determinar las dependencias iniciales y la comunicación entre servicios.

En esta actividad es donde el conjunto de servicios candidatos son identificados a partir de las entradas de los artefactos Modelo de Análisis del Negocio, los Objetivos del Negocio y Reglas del Negocio. El resultado es un Modelo de Servicio construido a un alto nivel que contiene un conjunto de artefactos como Particiones de Servicio y Especificaciones del Servicio.

Artefacto: Partición de Servicios.

Usadas extensivamente para desarrollar mapas y vistas de servicios particionándolos tanto por sus características técnicas como del negocio.

Artefacto: Colaboración de Servicio.

Este artefacto es utilizado para identificar servicios a partir del análisis de los roles y responsabilidades de la colaboración de los servicios candidatos.

3.4.3 Flujo: Diseño de Componentes.

El propósito de este detalle del flujo de trabajo es refinar el diseño del sistema.

3.4.3.1 Actividad: Diseño de Servicios.

La actividad Diseño de Servicios tiene como propósito:

- ✓ Definir los servicios y estructura de una solución orientada a servicios en términos de colaboraciones de elementos del diseño contenidos y interfaces/subsistemas externos
- ✓ Documentar la especificación del servicio.
- ✓ Determinar las dependencias y la comunicación entre servicios.

Durante esta actividad los servicios candidatos identificados en la actividad Identificar Servicios son refinados y desarrollados en modelos más detallados hasta llegar a conformar la propuesta final del Modelo de Servicios. El paso fundamental en la realización de esta actividad está en identificar a partir del Modelo de Servicios candidato, definido por el arquitecto del software durante la actividad Identificar Servicios, un modelo más específico al que se le adicionan los detalles del servicio, mensajes, entradas, canales, proveedores y las colaboraciones entre ellos.

Artefacto: Mensaje.

Durante el desarrollo de esta actividad aparece el diseño de mensajes Este artefacto es utilizado para describir los datos pasados dentro y fuera del servicio cuando se invocan las operaciones del mismo.

Mientras que el artefacto Colaboración de Servicios era utilizado anteriormente durante la actividad Identificar Servicios para investigar los roles y responsabilidades de los servicios, dentro de la actividad de Diseño de Servicios este artefacto puede ser utilizado para describir la implementación de un servicio como una colaboración (o coreografía) de un conjunto de servicios.

3.4.4 Flujo: Diseño de Componentes (Realización).

3.4.4.1 Actividad: Diseño de Subsistemas.

Esta actividad tiene como propósito

- ✓ Definir el comportamiento especificado en las interfaces de subsistemas en términos de colaboraciones de los elementos de diseño contenidos y interfaces/subsistemas externos
- ✓ Documentar la estructura interna del subsistema.
- ✓ Definir las relaciones entre las interfaces de subsistemas y las clases contenidas.
- ✓ Determinar las dependencias sobre otros subsistemas.

En el desarrollo de esta actividad se refina el diseño de servicios, identificando los subsistemas dentro del servicio. Durante la primera iteración de la fase de elaboración un componente puede actuar como una pequeña proyección del prototipo de la arquitectura para simular el comportamiento de un servicio. Más adelante este comportamiento es distribuido a una colaboración de clases contenidas dentro del subsistema del servicio.

3.4.5 Flujo: Diseño de BD.

El propósito de este detalle del flujo de trabajo es identificar las clases del diseño a ser persistentes en una base de datos y diseñar la estructura de la base de datos correspondiente.

El principal objetivo en la fase de elaboración es asegurarse de que el servicio encapsula todos sus requisitos de datos y que las dependencias entre servicios no surgen debido a la introducción de servicios que deben utilizar la misma fuente de datos.

3.5 Disciplina Implementación.

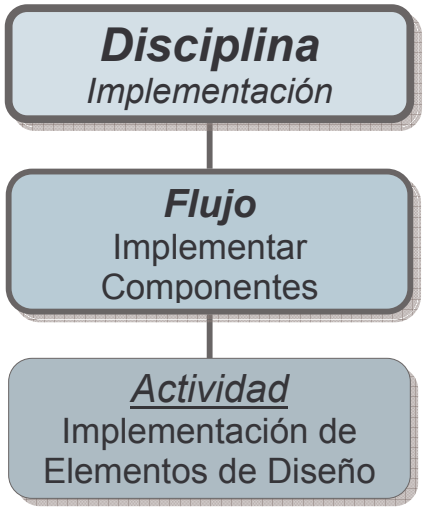


Ilustración 17: Disciplina Implementación.

El propósito de esta disciplina consiste en:

- ✓ Definir la organización del código en términos de subsistemas de implementación organizados en capas.
- ✓ Implementar los elementos del diseño en términos de elementos de implementación (binarios, ejecutables).
- ✓ Probar los elementos desarrollados como unidades.
- ✓ Integrar los resultados producidos por implementadores individuales (o equipos en un sistema ejecutable.

3.5.1 Flujo: Implementar Componentes.

El propósito de este detalle del flujo de trabajo es completar una parte de la implementación para poder entregarla para la integración al sistema.

Es importante enfatizar que el movimiento hacia el desarrollo de soluciones orientadas a servicios no invalida lo aprendido del desarrollo de soluciones basadas en componente, de hecho los componentes son usados como artefactos de implementación para los servicios. Esto se puede lograr de dos maneras.

- ✓ Del Modelo de Servicios, se generan los artefactos de servicios Web tales como el WSDL 1.1 el cual actúa como una descripción de la interfaz del servicio así como la información sobre la localización y los protocolos usados para tener acceso a una instancia del mismo. De estos artefactos la mayoría de las herramientas de diseño generan tanto elementos del servidor como accesorios del cliente para el servicio especificado.
- ✓ El Modelo de Servicios se puede transformar tanto para artefacto Modelo de Casos de Uso como para el artefacto Modelo de Diseño del Sistema. Los cuales pueden ser refinados más adelante para describir la implementación antes de generar tanto el artefacto Modelo de Implementación como la programación que representa la implementación del servicio.

En cualquier caso el acercamiento tomado durante la actividad Implementación de Elementos dependerán de la plataforma de tiempo de ejecución elegida.

Conclusiones.

El propósito de este capítulo fue a partir del proceso que describe el plug-in SOA de IBM, utilizar cada uno de los conceptos, actividades y nuevos artefactos que se generan, adaptarlo y configurarlo para las aplicaciones del Proyecto Atención Primaria de Salud.

CAPÍTULO 4 GUÍAS PARA LA EJECUCIÓN DEL PROCESO.

En este capítulo se explicarán una serie de guías para hacer más eficiente el trabajo con cada uno de los artefactos del Modelo de Servicios.

4.1 Guías.

Este proceso de desarrollo consta de varias guías donde se reflejan temas importantes para el modelado de aplicaciones con Arquitectura Orientada a Servicios. De forma general se incluyen guías para el Diseño de Mensaje y Mensaje Adjunto, Encapsulamiento del Servicio de Datos, Mediación de Servicio, Servicio y como Migrar de Servicios a Componente de Servicios.

4.1.1 Diseño del Mensaje y Mensaje Adjunto.

Adjunto del Mensaje.

Al diseñar mensajes, es común encontrar estructuras potencialmente grandes como documentos o cualquier tipo de archivo. Son muchos los archivos que pueden adjuntarse junto al mensaje que se envía, por ejemplo: informes o información de un producto determinado que se pueden distribuir como archivos PDF o un mensaje llevando consigo imágenes que describen un producto o un artículo.

El perfil de UML para los servicios de software proporciona un estereotipo adicional para tratar este tipo de mensajes: << *Adjuntos del mensaje* >> el cuál se puede unir a las características en un modelo de mensaje donde el contenido especificado está unido al modelo de cierta manera. Esto permitirá que el diseñador especifique detalladamente un aspecto muy importante del diseño de mensaje como es el caso del funcionamiento y el uso del ancho de banda, por el que pasan estos adjuntos.

Adjuntos o acoplamientos.

Hoy en día en Internet se está trabajando por lograr que las transferencias de información sean cada vez más grandes (en cuanto a cantidad de información) a través de URLs (**U**niform **R**esource **L**ocator) que permitan tener una localización común para que todos los clientes interesados o autorizados accedan a ella, permitiendo descargar la información por protocolos como el FTP (**F**ile **T**ransfer **P**rotocol).

Codificación del Adjunto.

El estereotipo del adjunto del mensaje posee una característica que denota la forma de codificación del adjunto, se recomienda que los valores que denotan la codificación para los adjuntos sean de tipo MIME (**M**ultipurpose **I**nternet **M**ail **E**xtensions). Estos tipos de codificación son utilizados en Internet por el protocolo http (**H**yper**T**ext **T**ransfer **P**rotocol) en la transferencia de datos binarios tales como imágenes para páginas Web.

4.1.2 Encapsulamiento del Servicio de Datos.

En esta guía se muestran dos puntos de vista que están muy relacionados: la noción de que un servicio debe hacer un encapsulamiento completo de los datos requeridos, y que la única vía de compartir la información entre los servicios se realiza mediante el intercambio del mensaje.

Servicios como fiefdoms.

Hoy en día cuando se habla de desarrollo orientado a objetos se escucha muy a menudo el término encapsulamiento o sea, la propiedad de que un objeto encapsule su estado (datos confidenciales) y su lógica de implementación. Este concepto se ha documentado, inicialmente en [HELLAND] y más recientemente en [SESSIONS] y fue centrado para el desarrollo de sistemas autónomos.

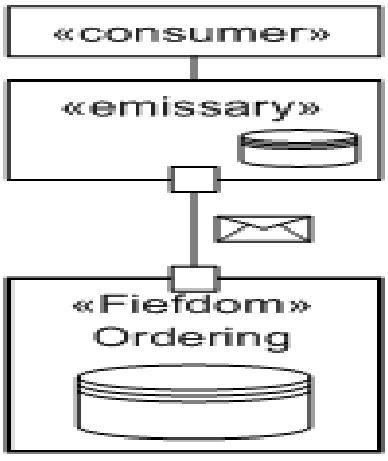


Ilustración 18: Servicios como fiefdoms.

La analogía de uso general [HELLAND] consiste en utilizar un agente. El agente actúa como el emisario a nombre del fiefdom de la compañía de seguros.

Papel del fiefdom.

El fiefdom es un servicio autónomo, que permite la comunicación a través de mensajes, estos mensajes son creados por los emisarios que actúan a nombre del consumidor. El fiefdom es seguro, autónomo y define totalmente un límite para los datos. No se pueden compartir fuentes de datos u otros datos persistentes entre los fiefdoms o entre ellos con otros elementos del software. Puede ser que un servidor de base de datos sostenga más de un servicio de persistencia pero es el accionar del fiefdoms el que asegura la integridad y seguridad de los datos.

Es importante que el agente pueda utilizar esta información, conociendo que esta información es:

a) una copia de los datos y no necesariamente los datos que el fiefdom está utilizando.

b) que la información esté fuera de tiempo y necesite ser actualizada.

Otro aspecto clave es asegurarse de que el fiefdom distribuya las copias de ciertos datos de referencia a los emisarios para que los almacenen y utilicen localmente y no afectar los datos originales. Por supuesto, es importante que el agente pueda utilizar esta información y que entienda que esta es una copia de los datos y no necesariamente de los datos que el fiefdom está utilizando, y que la información puede que no sea la más actualizada.

Las responsabilidades detalladas del fiefdom son:

- ✓ Manejar y distribuir los datos de referencia a todos los emisarios, identificando claramente las fechas de entrada de los datos.
- ✓ Manejar el estado de datos transaccionales, donde todas las transacciones se manejan enteramente en el interior.
- ✓ Validar toda la comunicación con los emisarios.

Papel del emisario.

El emisario actúa como un agente y puede ser situado uno por componente-consumidor que maneja los datos de referencia requeridos para completar los mensajes enviados de los procesos del fiefdom. Es también responsable de manejar copias locales de los mensajes por transacción, manteniendo una copia de la información hasta que se finalice la transacción.

En general se utiliza un emisario cuando la comunicación entre el fiefdom y el consumidor constituye una transacción más compleja que el emisario puede manejar con más eficiencia.

Las responsabilidades del emisario son:

- ✓ Actuar como agente a nombre del consumidor, terminar mensajes y obrar recíprocamente con el fiefdom.
- ✓ Manejar, cuando es apropiado, una transacción lógica de datos de referencia.
- ✓ Manejar una copia local de los datos de la referencia, que es actualizada a través de la decisión del fiefdom.

Límites de los datos de servicio.

En general hoy en día muchas aplicaciones son desarrolladas como sistemas de componentes integrados verticalmente. La forma más conocida para la integración de datos es contar con dos o más aplicaciones que compartan un mismo almacén de datos donde accedan a las mismas tablas dentro de una BD.

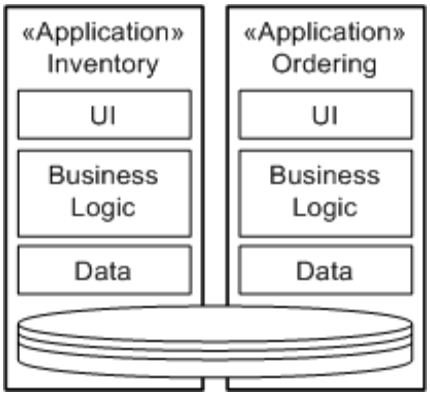


Ilustración 19: Datos manejados por dos servicios completamente autónomos.

Para el desarrollo de soluciones orientadas a servicios es recomendable que un servicio específico se encargue de manejar el límite de los datos dentro del modelo de datos como por ejemplo la Ilustración 19 donde los datos deben ser manejados por dos servicios completamente autónomos. Podría darse el caso de que existan elementos compartidos entre las dos aplicaciones lo cual trae consigo la creación de un tercer servicio que maneje los datos compartidos (hay que tener en cuenta que los datos compartidos deben tener un identificador que los diferencia de los demás elementos).

Obviamente el artefacto dominante en el análisis de los límites de los datos es el Modelo de Datos.

Mensajes de servicio como vista de datos.

Si todos los datos son almacenados solamente dentro del servicio y el acceso a los datos es restringido desde el exterior del mismo, entonces toda la comunicación tiene que ser a través de los mensajes identificados en la especificación del servicio. Estos mensajes representan copias de datos del servicio donde puede darse el caso que el dato sea una información antigua para el servicio.

Por ejemplo, en el caso de dos solicitudes casi concurrentes de un mismo dato, un mensaje puede devolver a un cliente que cuenta con 12 datos de una cantidad X, pero mientras esto ocurre otro consumidor del mismo servicio solicita 8 datos de los 12 de la cantidad X, lo cual puede traer consigo, que si el consumidor original intenta adquirir los 12 artículos de la cantidad X, la operación falle.

Los mensajes tienen que ser considerados como vistas y copias de datos.

4.1.3 Mediación de Servicio.

La mediación es el acto de la intervención entre partidos que están en conflicto para promover la reconciliación o compromiso. Las formas más comunes de reconciliación se encuentran en los sistemas distribuidos en general y en las soluciones orientadas a servicios en particular.

- ✓ **Mediación de la interfaz:** es el cambio entre las definiciones de la operación entre el remitente y el receptor.

- ✓ **Mediación del protocolo:** la mayoría de las soluciones basadas en componentes giran alrededor de los protocolos de un campo común (o sistema de campo común) para la comunicación. La comunicación entre los mensajes de servicios tiene que atravesar por diferentes protocolos entre el remitente y el receptor.
- ✓ **Mediación de la operación:** esta forma de mediación puede parecer familiar para los desarrolladores y se relaciona con el patrón de estrategia. Un componente puede básicamente seleccionar en un sistema de implementación un servicio en particular, o seleccionar la operación basada en parámetros runtime, o el contenido de la petición (esto también se conoce en ocasiones como encaminamiento basado en contenido).

Es importante apuntar que más y más plataformas del middleware están proporcionando capacidades para la mediación avanzada sin tener que desarrollar componentes explícitos de la mediación. Estas plataformas pueden proporcionar a los mediadores la posibilidad de seleccionar la implementación correcta de una petición X de un consumidor, basada en: el contenido del mensaje y las reglas de negocio.

La mediación de los datos en actividades.

En el caso de que los mensajes requieran de una transformación entre el remitente y el receptor, es posible utilizar la capacidad para denotar la transformación entre el remitente y el receptor proporcionada por el lenguaje UML 2.0.

La transformación es un elemento reutilizable que se puede identificar para transformar un tipo de mensaje a otro, o para mediar entre un servicio que se envía y otro que se recibe.

La mediación de los datos se puede también tratar como un patrón concreto de la iteración del servicio, es decir, que existe un servicio explícito de la mediación que es responsable de implementación de una o más transformaciones de los datos. En este caso el mediador tiene que responder a los mensajes enviados por el consumidor, transformar el mensaje y pasarlo hacia el servicio, como se muestra en la Ilustración 20:

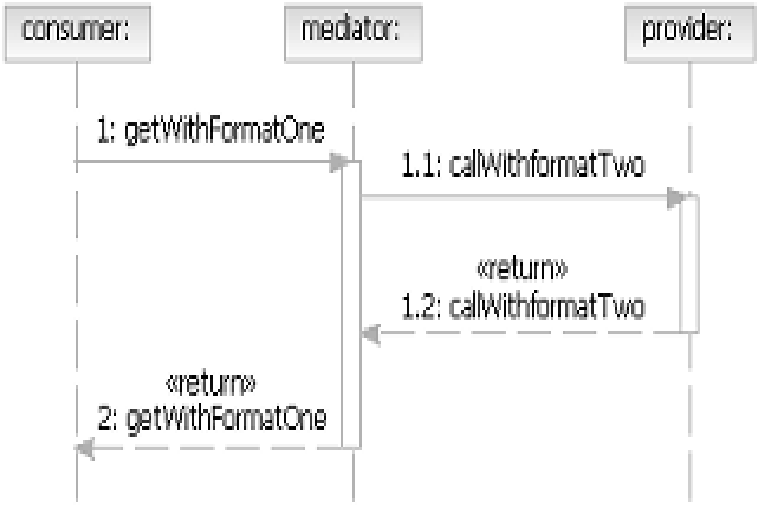


Ilustración 20: Mediación de los datos.

La mediación del protocolo en la mediación de las entradas.

La información del protocolo se especifica para el atascamiento de un canal de servicios donde es posible introducir elementos adicionales que alteran la especificación del protocolo. Por ejemplo, en el siguiente diagrama de estructura (Ilustración 21) se muestran dos particiones, una para los servicios de enfrentamiento de la Web y una para los servicios internos: existe además un canal de servicios entre las particiones con un atascamiento de "HTTP-SOAP (**S**imple **O**bject **A**ccess **P**rotocol)" común para los servicios de enfrentamiento de la Web (Web facing services).

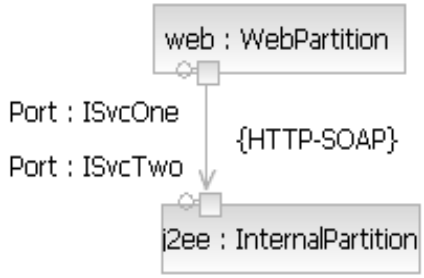


Ilustración 21: Diagrama de estructura de dos particiones.

La idea principal es apoyar el nivel requerido de funcionamiento (y de otros requisitos no funcionales) para que toda la comunicación dentro de la partición interna ocurra sobre los protocolos específicos de la plataforma. La Ilustración 22 muestra cómo un servicio está conectado a la entrada del servicio. "Port: ISvcTwo" utilizando el protocolo de Java RMI (**J**ava **R**emote **M**ethod **I**nvocation).

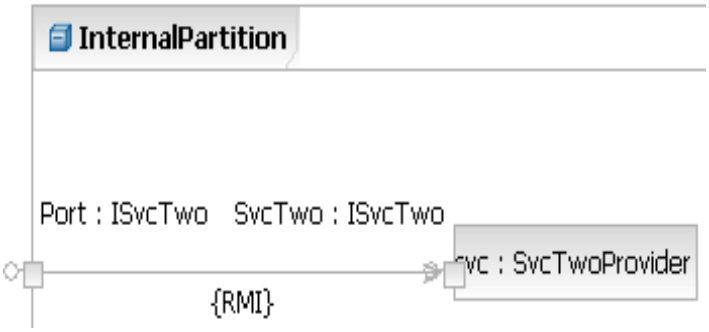


Ilustración 22: Conexión entre un servicio y la entrada de un servicio.

Pero ¿cómo la partición WEB conecta con la misma entrada usando HTTP-SOAP?

La respuesta es que, la entrada del servicio puede mediar el protocolo, convirtiendo las estructuras del mensaje y las invocaciones de un formato a otro. Ésta es una funcionalidad común proporcionada generalmente por middleware como Object Request Brokers (ORBs) o Message Brokers, (corredores de la petición del objeto (orbes) o corredores del mensaje), y que puede describirse "como un servicio que toma llamadas de la partición WEB y las vuelve a enviar a servicios incluidos.

Mediación de la invocación usando la composición del servicio.

Como se describe anteriormente es común definir una estructura donde existan dependencias entre los servicios para realizar una operación, donde el consumidor solicite uno o varios de estos servicios siguiendo las reglas del negocio aplicables a la información que se solicita para que el servicio elija una respuesta basada en las necesidades del cliente. Un ejemplo común es que para una petición de un cliente determinado, el servicio de recepción pueda elegir una o dos implementaciones basadas en el nivel de preferencia del cliente, como podemos observar en la Ilustración 23.

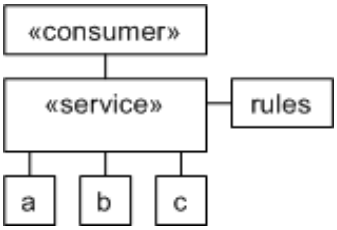


Ilustración 23: Dependencias de los servicios.

Es posible construir la solución como sistema de servicios, donde el mediador, las reglas y todos los implementadores se representan por servicios separados. Lo cual puede comprobarse en la Ilustración 24:

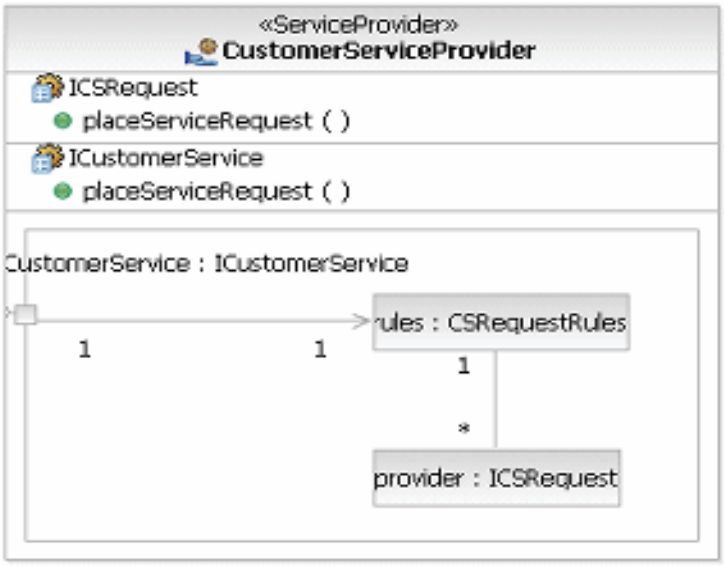


Ilustración 24: Estructura compuesta del servicio.

El componente de la mediación posee también una especificación del servicio requerida para ser implementado por todos los servicios que media. Esto permite definir la estructura compuesta del servicio (ilustración anterior) y el comportamiento dinámico que se muestra en la Ilustración 25, donde puede observarse que la representación de los servicios mediados es denotada por la interfaz requerida y demostrada con una multiplicidad ilimitada.

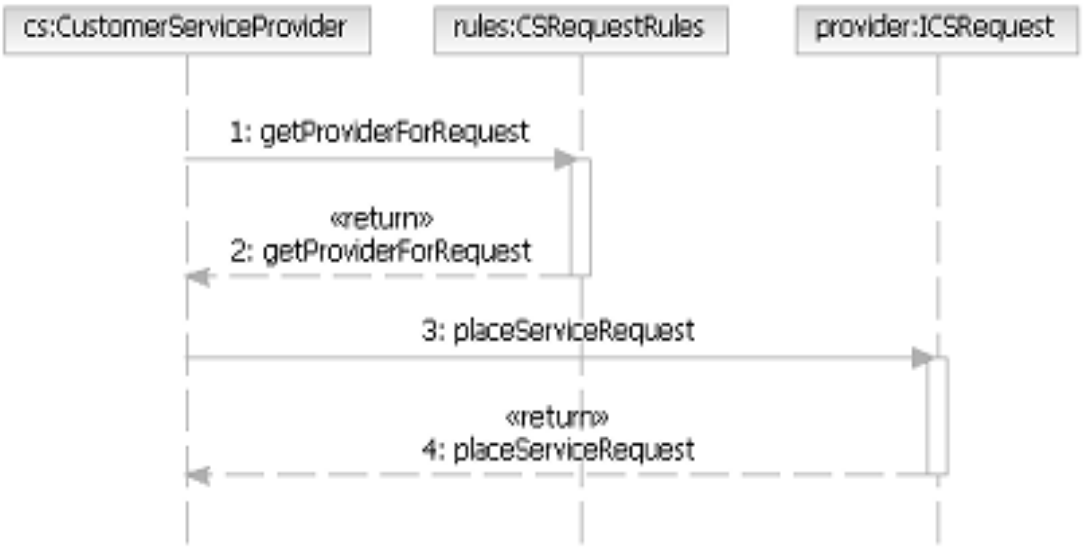


Ilustración 25: Comportamiento dinámico de un servicio.

Este tipo de ruta basada en contenido o basada en las reglas del mensaje puede ser culminada por la plataforma del middleware, elegida como parte de la arquitectura de la solución.

4.1.4 Servicio.

Un servicio es obviamente el artefacto dominante en una Arquitectura Orientada a Servicios, pero ¿que es un servicio? RUP plantea la siguiente definición:

Un servicio es un recurso del software (descubrible) que exterioriza la especificación (Externamente Especificado) del propio servicio. Esta especificación del servicio está disponible para buscar e invocar al consumidor de servicios. El Proveedor de Servicios realiza la implementación de la especificación del servicio y también entrega y asegura la calidad de los requisitos del servicio al consumidor de servicios. Los servicios serán gobernados por políticas declarativas, apoyando así un estilo arquitectónico dinámicamente re-configurable.[12]

Descubrible.

[12] **RUP, P. U. D. S., 2005.**

Un servicio descubrible debe proporcionar un sistema metadata (compañía productora de software para aplicaciones on-line e industriales) que permita la clasificación del servicio y su especificación externa (la clasificación como parte de la especificación).

Externamente especificado.

La especificación externa permite que un servicio publique sus detalles (interfaz, localización, políticas, clasificaciones) sin la necesidad de que un cliente tenga acceso al servicio. Tal información entonces se almacena generalmente en una localización conocida o un registro especializado del servicio que soporte las preguntas del sistema metadata. Actualmente en el mundo de los servicios Web el estándar aceptado para la descripción de las interfaces del servicio es WSDL (idioma descriptivo de los servicios Web del consorcio mundial Web).

Contrato base.

La especificación del servicio proporciona una visión tanto para el proveedor como el consumidor de servicios.

La Tabla 17 describe los diversos aspectos de una especificación del servicio que efectúan el proveedor y el consumidor de la especificación.

Rol	Especificación de interfaz	Especificación del comportamiento	Especificación de la política
Proveedor	Informa al sistema acerca de las operaciones y los mensajes que el servicio debe responder, donde todas las operaciones deben responder y contestar los mensajes correctos.	Informa el comportamiento que este servicio debe apoyar. Si tal especificación del comportamiento es formal y completa, entonces puede probarse una implementación para la completa conformidad de la	Informa en cuanto a un sistema de restricciones cuál implementación del servicio puede trabajar correctamente como un sistema de cualidades del servicio.

		especificación.	
Consumidor	Informa al sistema las operaciones que pueden ser invocadas.	Informa a los requisitos de protocolo lo que el consumidor debe realizar (ordenar la operación, los flujos de datos, etc.). También indica cualquier operación que el consumidor deba poner en ejecución para apoyar la colaboración.	Informa acerca de los requisitos de seguridad para comunicarse con este servicio. También identifica la calidad del servicio que un consumidor puede conseguir de un Proveedor de Servicios dado.

Tabla 17: Aspectos de una especificación del servicio efectuada por los proveedores y consumidores de la especificación.

Alineación del negocio.

Puede observarse una conexión cercana entre el modelo de negocio y los servicios asociados, ya que las operaciones suministradas por los servicios en ocasiones pueden ser identificadas en el nivel del negocio, lo cual significa que las tareas identificadas en el proceso de modelado del negocio pueden ser directamente realizadas como operaciones del servicio por lo que los usuarios de las aplicaciones se convierten en parte directa de los procesos de análisis y diseño.

Modelado de un servicio.

El modelado del servicio utiliza el perfil de UML 2.0 para los servicios de software y la dirección previa de cada elemento en el perfil. En el Ilustración 26 se muestra la visión estática de los servicios y de las especificaciones del servicio en un Modelo de Servicios:

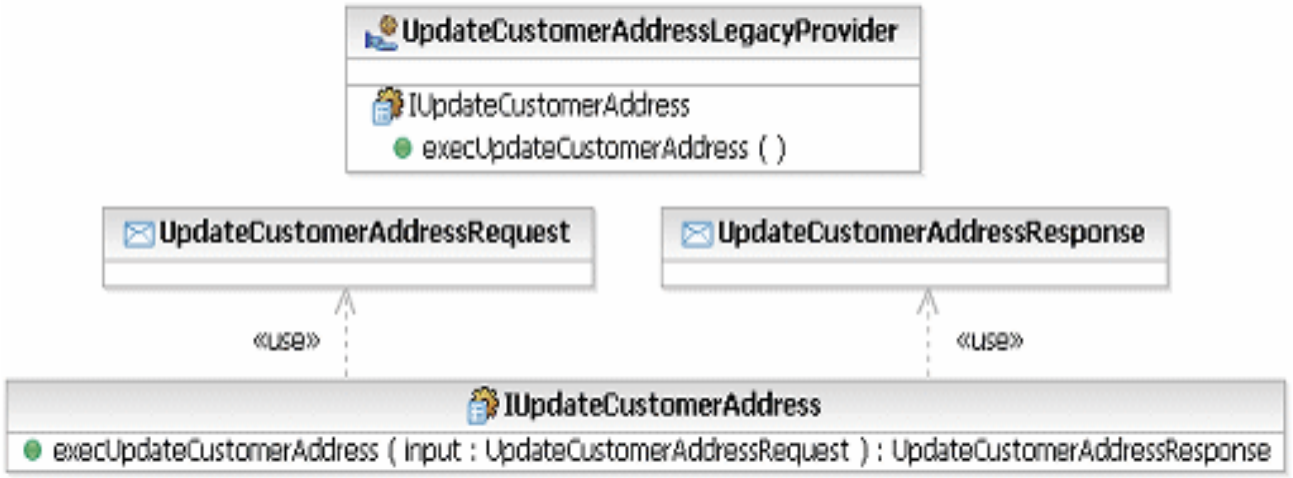


Ilustración 26: Visión estática de los servicios y de las especificaciones del servicio en un Modelo de Servicios

- ✓ El proveedor de servicios "UpdateCustomerAddressLegacyProvider" proporciona un servicio con la “dirección del cliente actualizada”.
- ✓ El servicio "UpdateCustomerAddress" pone la especificación "IUpdateCustomerAddress".
- ✓ La especificación "IUpdateCustomerAddress" tiene una sola operación "execUpdateCustomerAddress".
- ✓ La operación "execUpdateCustomerAddress" toma un solo mensaje de entrada, "UpdateCustomerRequest".
- ✓ La operación "execUpdateCustomerAddress" devuelve un solo mensaje de salida, "UpdateCustomerResponse".

4.1.5 Migrar de Servicios a Componentes de Servicios.

El propósito del Modelo de Servicios es sobre todo identificar los servicios, su organización, la colaboración y una especificación detallada y completa de cada uno de ellos. En general el Modelo de Servicios es controlado por el Modelo de Diseño, no obstante, es posible generar los artefactos directamente del Modelo de Servicios de acuerdo a las especificaciones que requiera el servicio, o puede crearse un Modelo de Casos de Uso a partir del Modelo de Servicios que permita la descripción adicional de los servicios antes de su construcción. La ventaja de estas opciones es mostrada en la Ilustración 27

donde a partir del Modelo de Servicios se realiza una conexión con el Modelo de Casos de Uso, con un Modelo de Diseño y luego a la implementación.

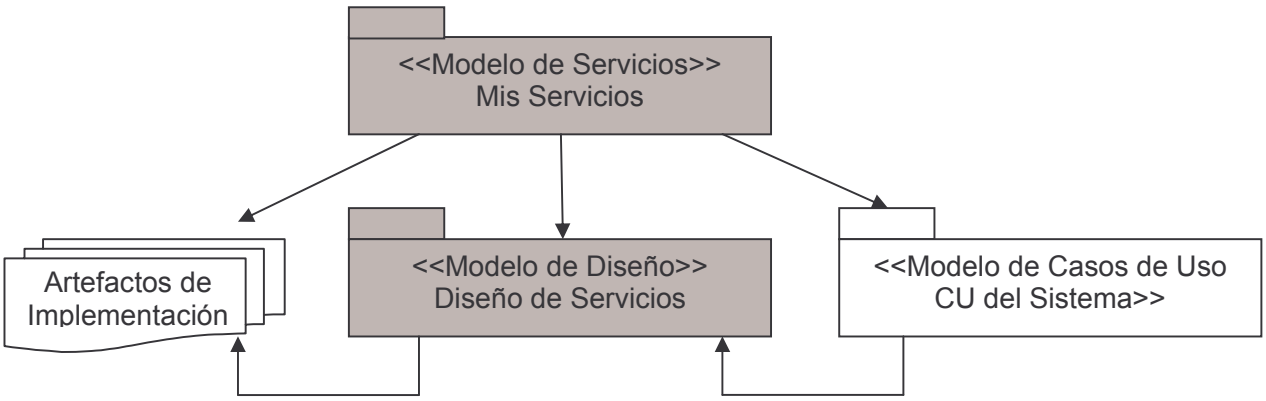


Ilustración 27: A partir del Modelo de Servicios conexión con el Modelo de Casos de Uso, con el Modelo de Diseño y luego a la implementación.

Como fue mencionado anteriormente es recomendable que el Modelo de Servicios sea controlado utilizando un Modelo de Diseño, propiciando la posibilidad al diseñador y al implementador de poder aplicar patrones al Modelo de Diseño para modelar las capacidades y estructuras adicionales ante la generación de los artefactos de la fase de Implementación.

Creando el componente de servicios.

La implementación real de un servicio o de un sistema de servicios tiene lugar a través de la realización de una especificación del servicio por un componente de servicios. La especificación del servicio proporciona el contrato de implementación, donde la tecnología o las técnicas usadas para poner el servicio en ejecución son inaplicables mientras se satisface el contrato. En la Ilustración 28 se muestra la relación entre los servicios que identificados y los componentes y objetos que proporcionan la implementación de estos servicios.

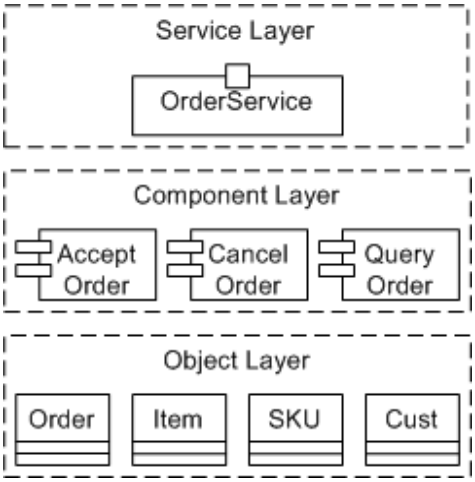


Ilustración 28: Relación entre los servicios, componentes y objetos.

De esta manera puede observarse cómo el Modelo de Diseño de RUP se puede utilizar para capturar el diseño de las capas del componente y del objeto. Un aspecto importante de la relación entre el Modelo de Servicios y el modelo de componentes del diseño es que el sistema de especificaciones del servicio representa los contratos que deben ser satisfechos, las operaciones identificadas en las especificaciones que deben ser implementadas, los consumidores de servicios que utilizan este mismo modelo para entender la interfaz y el comportamiento de los servicios que esperan utilizar.

La Ilustración 29 representa un proveedor de servicios.

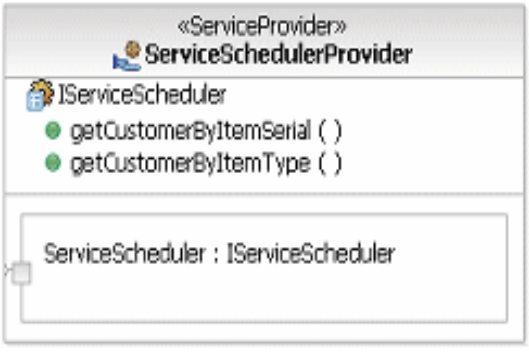


Ilustración 29: Representación de un proveedor de servicios.

La llave para la utilización del componente de servicios debe ser directamente detectable en el Modelo de Servicios, y la manera más fácil de lograr esto es mediante el elemento de la especificación del servicio (una interfaz de UML) que puede ser controlada a través del componente de servicios, consiguiendo el siguiente resultado:

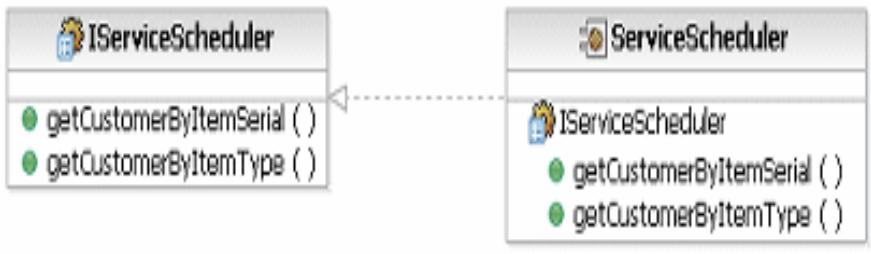


Ilustración 30: Control de una especificación del servicio a través del componente de servicios.

Definir un sistema de componentes y las clases que proporcionan el comportamiento del componente que resulta pasa a ser parte de la responsabilidad del implementador, a partir de lo planteado en la Ilustración 30.

Patrones de diseño de componente de servicios.

Decir que un componente de servicios simplemente realiza la especificación del servicio no le brinda al implementador una gran ayuda para ir desde la definición de alto nivel del servicio a un conjunto de implementaciones de bajo nivel, para ello se requiere de un conjunto de clases y artefactos mediante los cuales se proporciona el comportamiento del servicio. Con relación a esto es usual contar con un conjunto de patrones que guíen las políticas de los requerimientos, a continuación mencionamos algunos patrones que son útiles en el diseño del componente de servicios.

El patrón Componente de Servicios Bajo.

En la definición inicial de la estructura de un servicio, como se muestra en la Ilustración 31, el siguiente patrón brinda un punto de partida para la configuración y la finalización del mismo.

Ilustración 31: Patrón componente de servicios bajo

Los elementos del patrón son:

Componente fachada: la fachada realiza ella misma la interfaz del componente de servicios y proporciona las capacidades básicas para la validación del mensaje antes de pasar la petición de ejecución al componente por-operación. En este caso se estereotipa el componente como <<facade>> para una mejor claridad.

Componente por-operación: brindar un servicio granular es útil en la mayoría de los casos para tener separada en componente y/o clase la implementación de cada una de las operaciones brindadas por el servicio

En la Ilustración 32 se muestra la estructura de este patrón, en este caso la fachada es designada por el propio componente de servicios y por el consumidor para solicitar las operaciones del componente de servicios, esto puede ser modelado utilizando el UML 2.0 mediante el cual se expone las interfaces y se hace la designación explicita usando conectores.

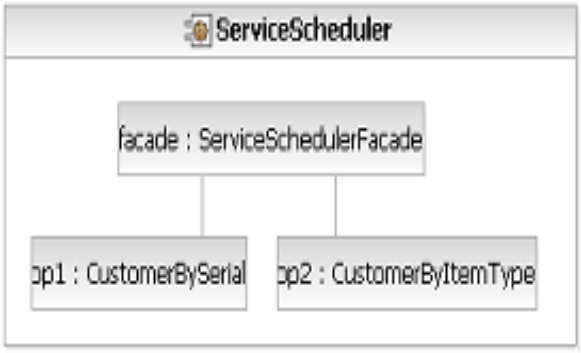


Ilustración 32: Estructura del patrón componente de servicios bajo

Patrón: Operación Simple.

En algunos casos donde los servicios son identificados en el Modelo de Servicios con operaciones múltiples, es más apropiado implementar las operaciones individualmente como servicios independientes separando las vistas lógica y física del servicio. Tal patrón tiene ventajas en términos de la flexibilidad del contrato, de la alta disponibilidad, del control de versiones y de la evolución, pero pierde la noción de una interfaz de un servicio como un conjunto de operaciones relacionadas.

Modelando los componentes de servicios según este patrón tenemos un simple <<componente de servicios>> realizando una simple interfaz con una simple operación.

En este caso, no hay realización directa de la especificación original del servicio por ningún elemento; esto es una fortaleza por lo que introduce un elemento en el modelo que brinda la trazabilidad hacia la especificación del servicio (hacia atrás). En la Ilustración 33 puede observarse el componente estereotipado como <<subsystem>> el cual es la implementación de la especificación del servicio y propietario de los elementos descritos.

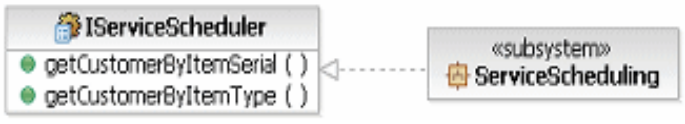


Ilustración 33: Componente de servicios como subsistema.

Este patrón no introduce el componente <<facade>> pues el consumidor de los servicios es el responsable de identificar los servicios que él utiliza.

Patrón: Operación Mediadora.

Si existe la posibilidad de que una solicitud de un consumidor de un servicio sea enrutada para la ejecución de una operación de un componente es posible extender el patrón con un mediador que enrute los mensajes, en la Ilustración 34 se muestra un ejemplo. El mecanismo exacto usado por el mediador no está definido y no deberá ser utilizado con el patrón operación simple.

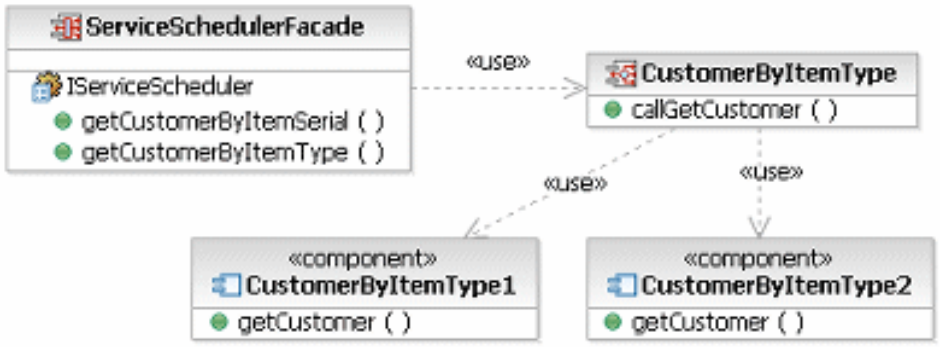


Ilustración 34:Patrón Operación Mediadora.

En la Ilustración 35 se ve la estructura del patrón, la conexión del mediador es certificada desde la fachada (<<facade>>) la cual la usa directamente para llamar las operaciones del componente.

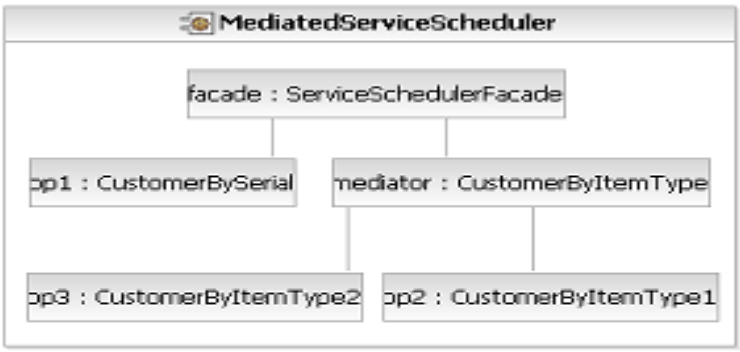


Ilustración 35: Estructura del Patrón Operación Mediadora.

Componente de acceso a datos.

Siempre que las operaciones de un servicio compartan iguales requerimientos de datos es útil destinar un componente en específico que brinde, para la implementación, las capacidades de administración de datos.

Conclusiones.

El propósito de este capítulo fue proponer un proceso formal y eficiente para el empleo de los elementos que componen el artefacto Modelo de Servicios. Se considera que lo desarrollado en el capítulo sirve como guía y ayuda para la ejecución del proceso propuesto.

CONCLUSIONES.

Con la realización del presente Trabajo de Diploma se arribaron a las siguientes conclusiones:

- ✓ Se realizó un detallado estudio sobre el contenido del Plug-in de RUP para SOA.
- ✓ Se describió el proceso de desarrollo, lo cual incluyó:
    - Fases y Disciplinas afectadas.
    - Actividades de cada Fase y Disciplina.
    - Artefactos generados de cada actividad.
    - Trabajadores y Responsabilidades dentro de cada Fase y Disciplina.
- ✓ Se configuró el proceso de desarrollo de la metodología RUP para la Arquitectura Orientada a Servicios en las aplicaciones del Proyecto Atención Primaria de Salud.

Con los resultados del estudio realizado y la configuración del proceso de desarrollo para aplicaciones con arquitectura SOA se cumple cabalmente con el objetivo general propuesto, ya que se obtuvo un procedimiento para el desarrollo de las aplicaciones sobre Arquitectura Orientada a Servicios en el Proyecto Atención Primaria de Salud siguiendo las extensiones de la metodología RUP.

RECOMENDACIONES.

Poner en práctica este proceso de desarrollo para aplicaciones con Arquitectura Orientada a Servicios en las futuras aplicaciones del proyecto APS.

Realizar un estudio profundo de la herramienta Rational Software Architect, para ejecutar todo el proceso desarrollado y diseñar el Modelo de Servicios.

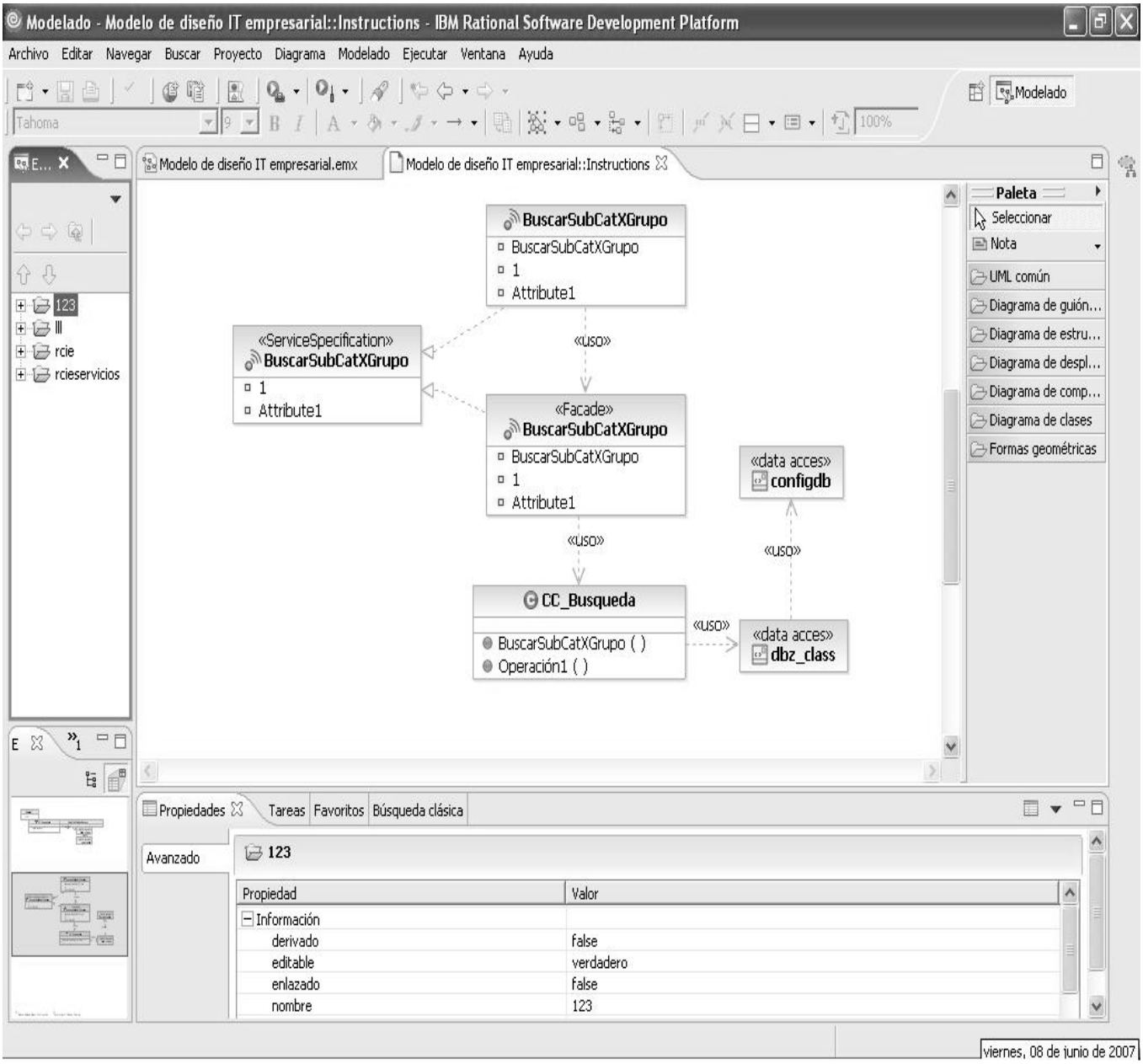
BIBLIOGRAFÍA.

1. Ayuda del Rational.
2. Brey, Gustavo A. (2006). *SOA (Arquitectura Orientada a Servicios)*. Consultado en Enero 22, 2007 en http://www.ibm.com/es/.
3. Clements, P, et al. (September 27, 2002). *Documenting Software Architectures: Views and Beyond.* : Addison Wesley, ISBN: 0-201-70372-6.
4. Desalvo, Claudio. (2007). SOA - Un Paradigma de Arquitectura Corporativa. Consultado en Enero 21, 2007 en http://www.epidataconsulting.com.
5. Everis (2007). Arquitecturas Empresariales. Orientación a Servicios (SOA) y Gestión de Procesos de Negocio (BPM). Consultado en http://www.aulesempresa.upc.edu/programes/everis.pdf.
6. http://es.wikipedia.org/wiki/Portada
7. Gómez, A. (2007). Partamos con algo más técnico. Consultado en http://alvarogomezrubio.blogspot.com/search/label/SOA.
8. Mendoza, María A. (2004). *Metodologías De Desarrollo De Software.* Consultado en Febrero 6, 2007 en http://www.informatizate.net.
9. MSF, metodología aplicada. willydev.net. Consultado en http://www.willydev.net/descargas/articulos/general/MSF.aspx.
10. Parra, J. (2007). Hacia una Arquitectura Empresarial basada en Servicios. MTJ.NET. Consultado en 03, 15,2007 en http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp.
11. Pellegrini, C. (2007). SOA: Service Oriented Architecture. Consultado en http://www.itcon.com.ar/itcon/Miscelaneos-SOA.
12. Potosí, S. (2005). Servicios y Soluciones. Stream Nova. Consultado en http://www.streamnova.com/servicios.htm.
13. Rational Unified Process (RUP). Disponible en: https://pid.dsic.upv.es/C1/Material/Documentos Disponibles/Introducci%C3%B3n a RUP.doc
14. Reyes, P. (2002). Una nueva revolución MDA. Consultado en http://72.14.209.104/search?q=cache:zFRJ4A1jjxwJ:www.danysoft.info/free/mdarev.pdf+metodologia+MDA&hl=es&ct=clnk&cd=12&gl=cu.

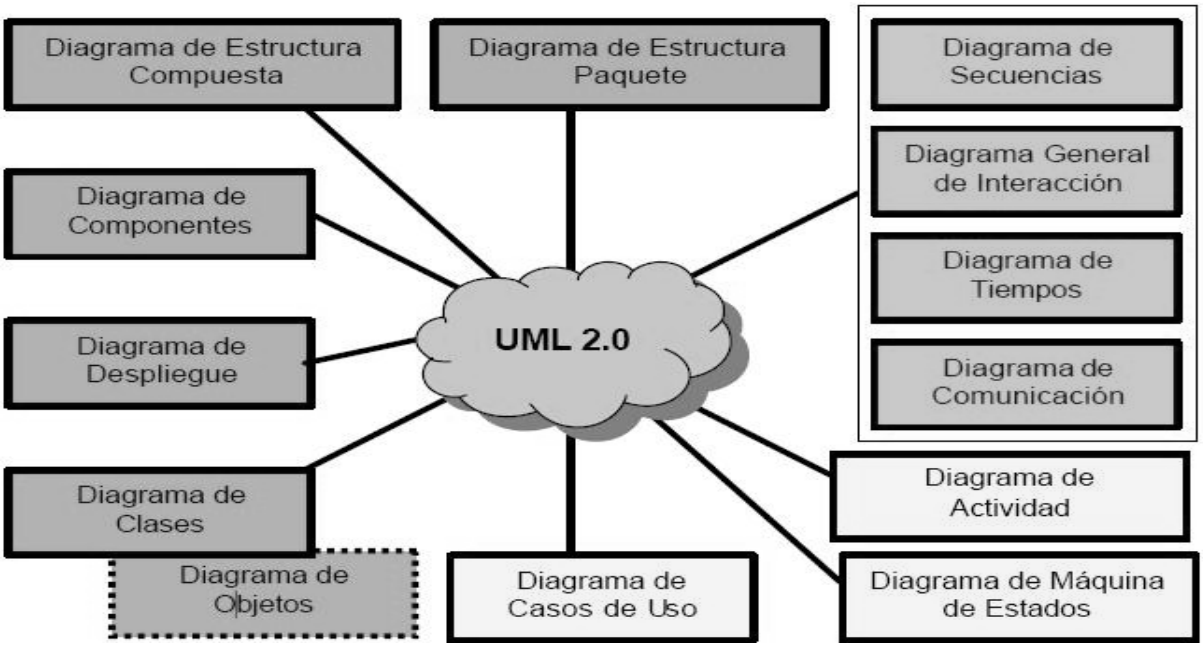
15. Salm, J. (2007). Arquitectura SOA: Uso eficiente de Web Services, WSDL e UDDI. Consultado en http://bvs4.icml9.org/gt/ti/activity.php?lang=es&id=3.
16. Sánchez, Diana M., Cavero, José M., Marcos, Esperanza. (2005). *Ontologías y MDA: una revisión de la literatura*. Consultado en Febrero 6, 2007 en http://www.omg.org/docs/omg/.
17. Shah, R. (October 19, 2005). *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*. : IBM Press, ISBN: 0-13-187002-5.
18. wigahluk (2007). Percepciones acerca de la programación ágil y XP. Consultado en http://wigahluk.wordpress.com/2007/06/14/percepciones-acerca-de-la-programacion-agil-y-xp/.
19. (2005). Computación Grid. Consultado en http://www.textoscientificos.com/redes/computacion-grid.
20. (2005). SOA components. Consultado en http://www.morfeo-project.org/index.php?option=com_content&task=view&id=50&Itemid=78.

ANEXOS.

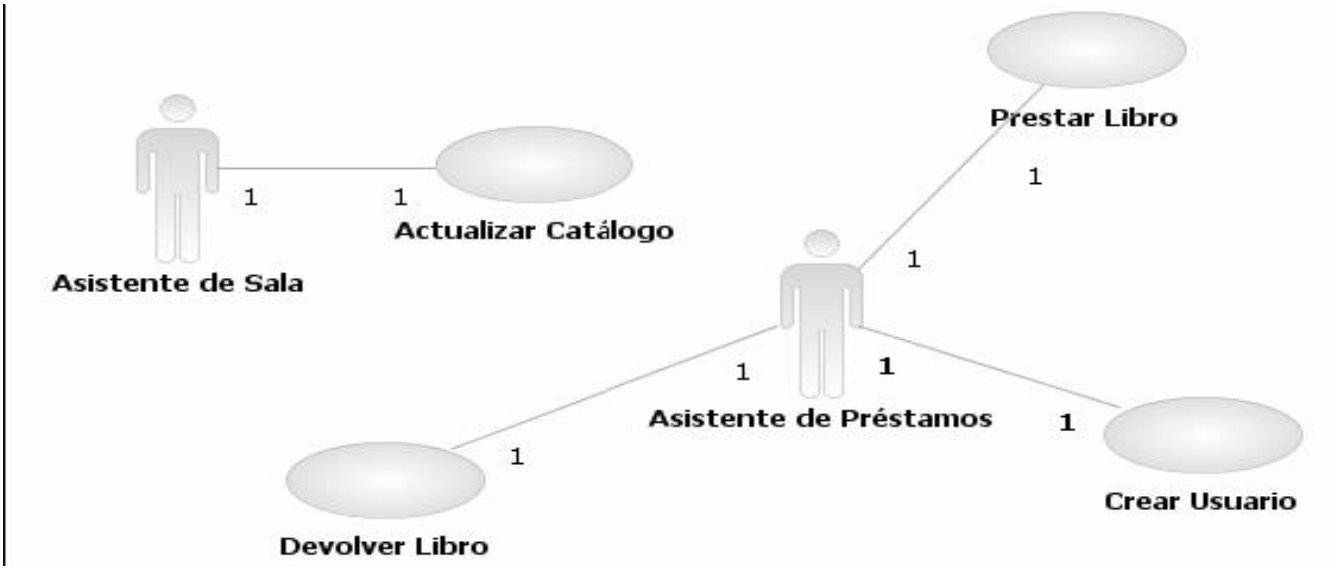
Anexo 1: Rational Software Architect.



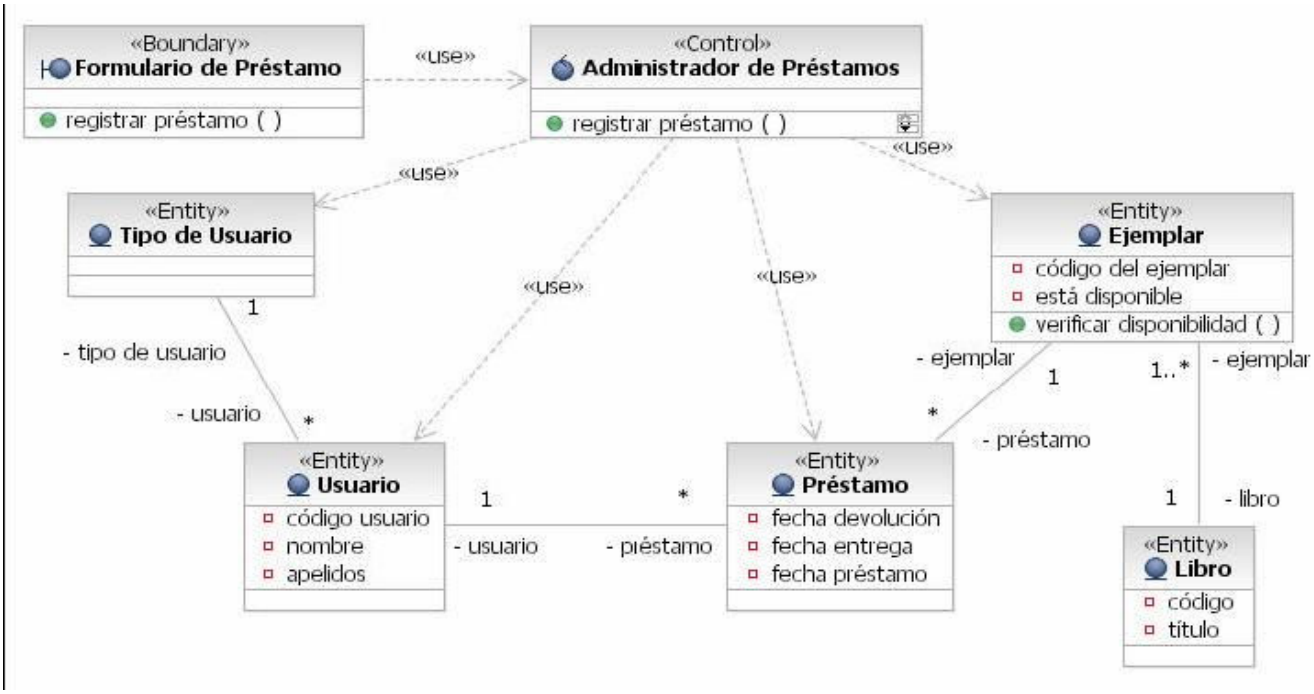
Anexo 2: UML 2.0.



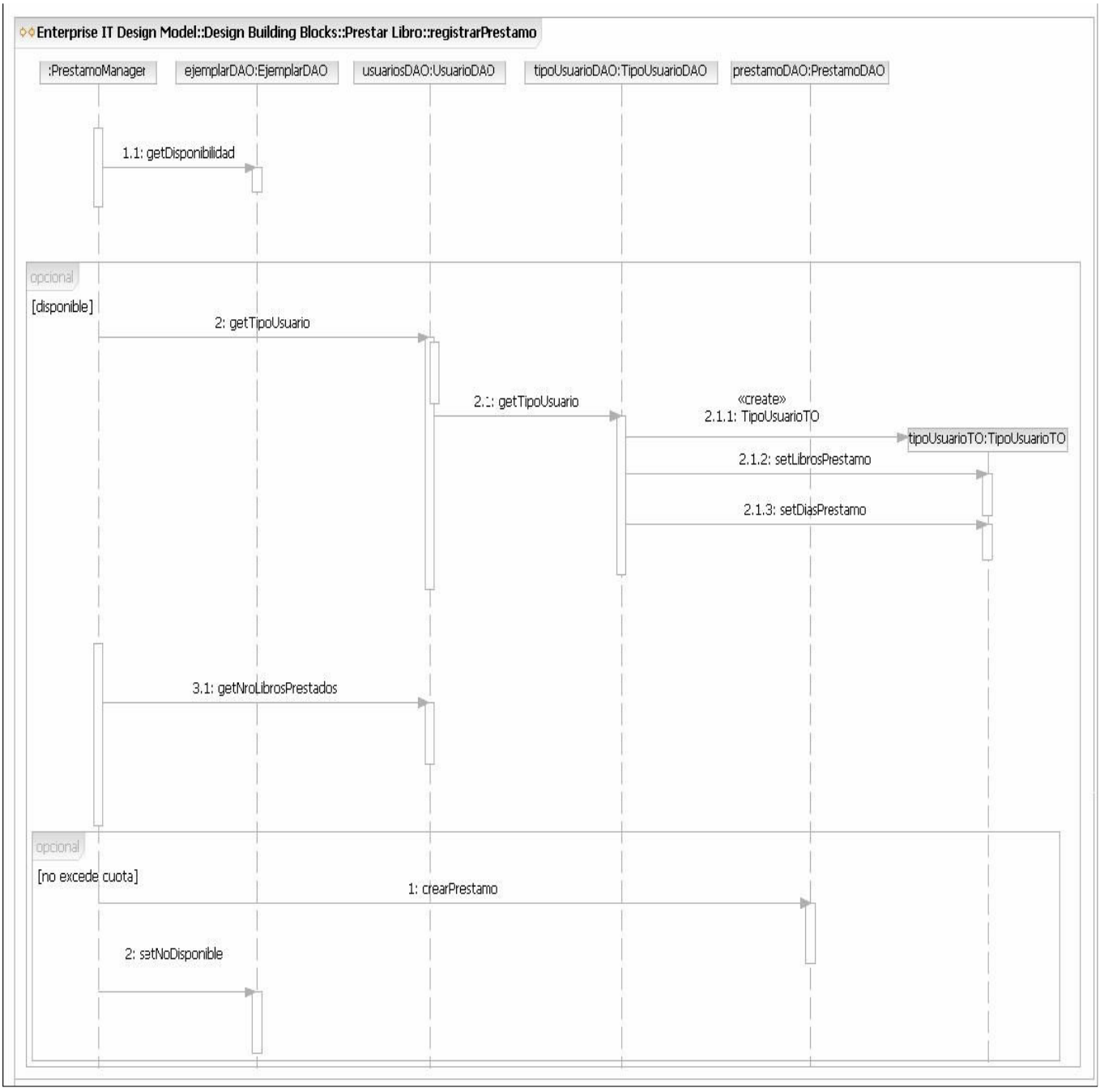
Anexo 3: Diagrama de Casos de Uso.



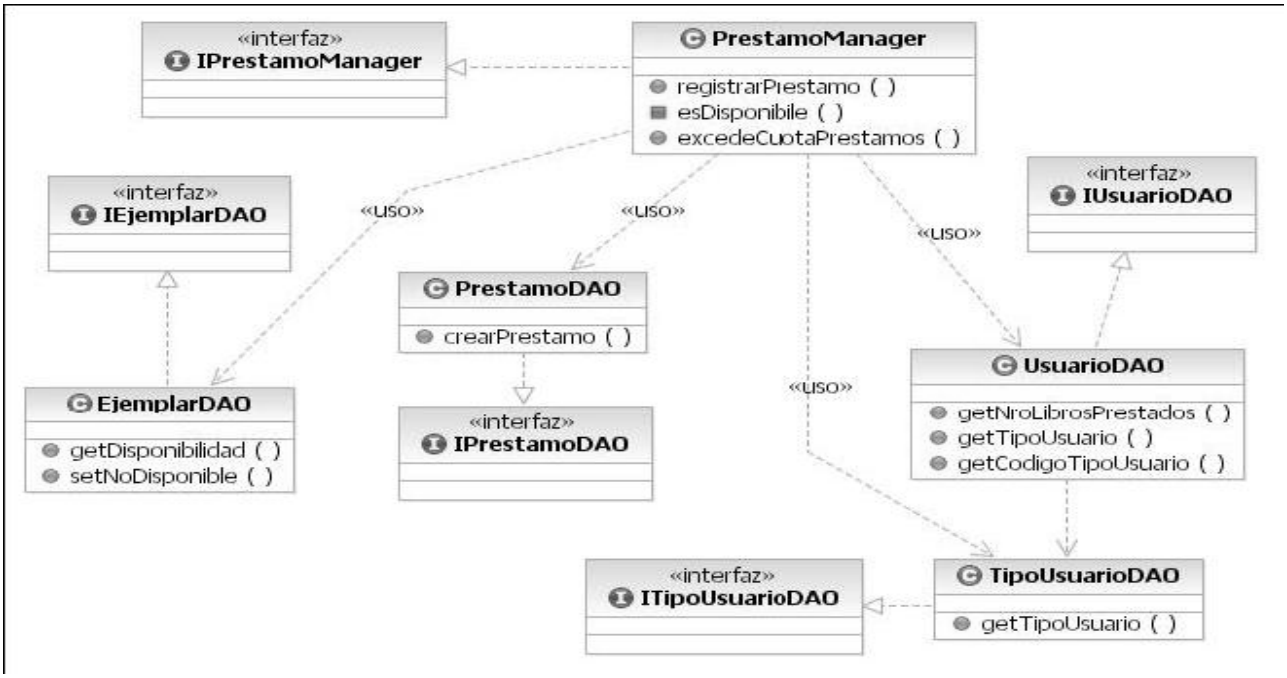
Anexo 4: Diagrama de Clases del Análisis.



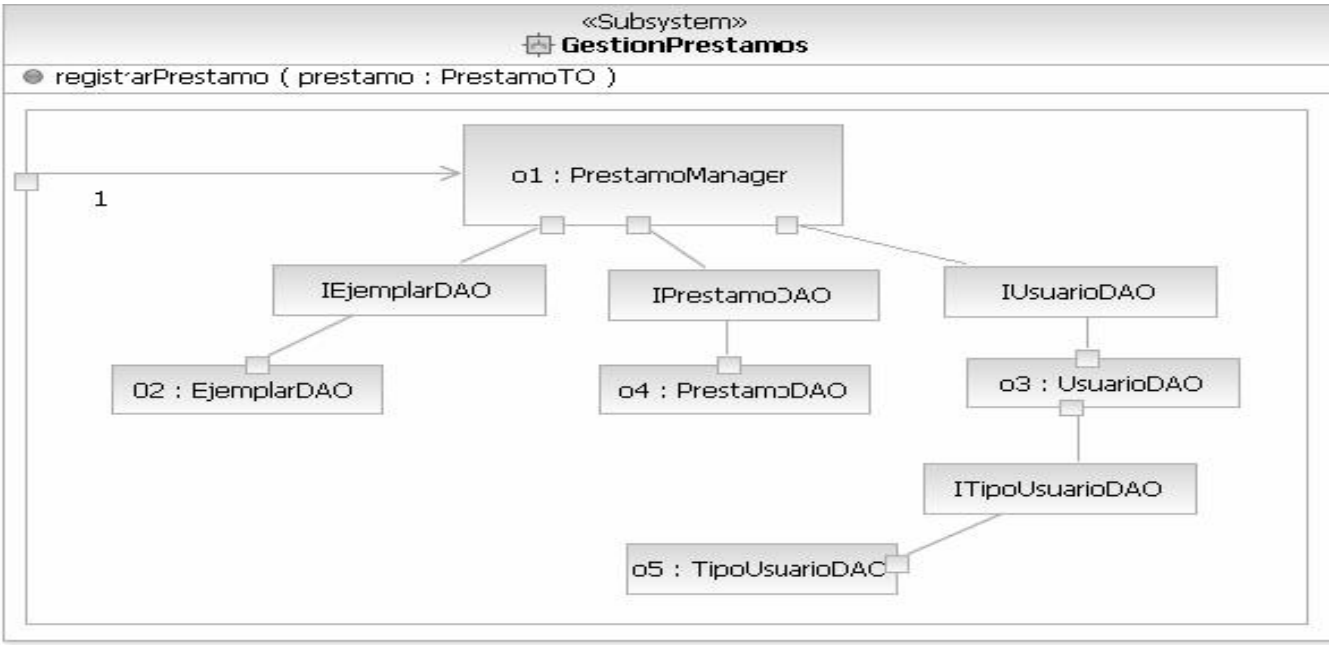
Anexo 5: Diagrama de Secuencia.



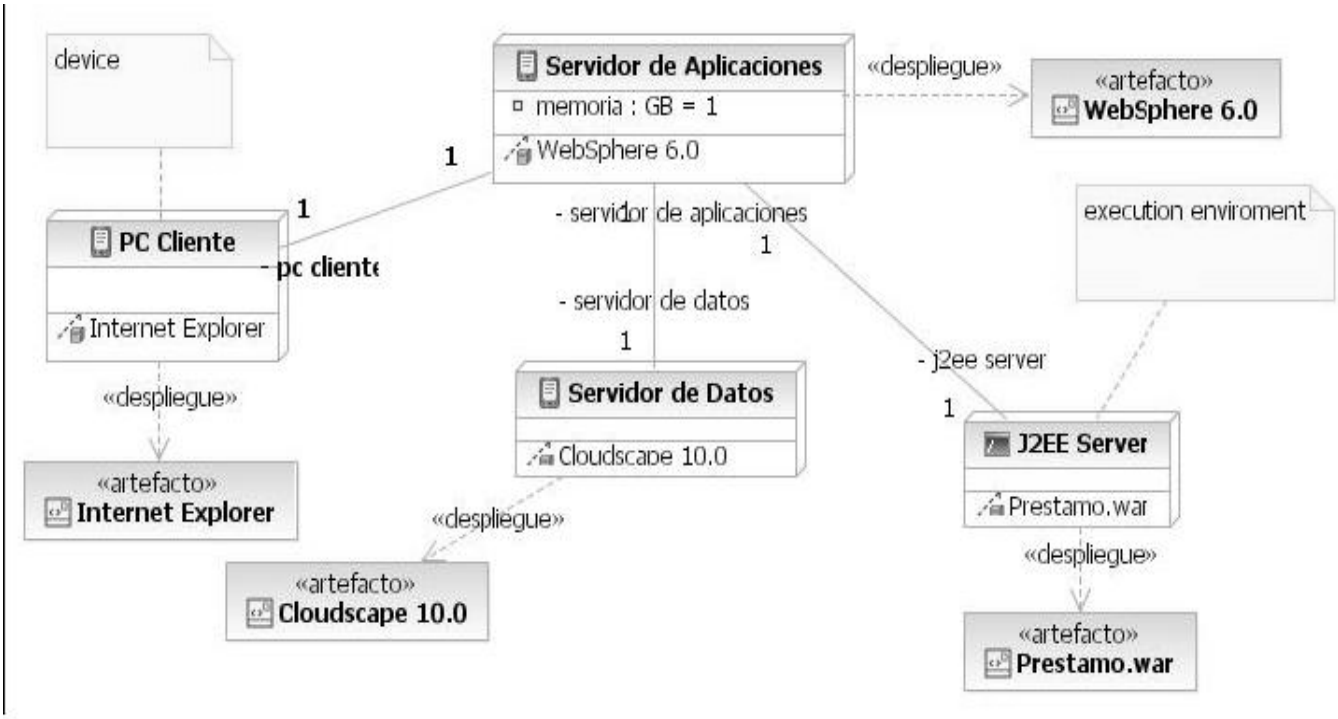
Anexo 6: Diagrama de Clases del Diseño.



Anexo 7: Diagrama de Componentes.



Anexo 8: Diagrama de Despliegue.



GLOSARIO.

AES: En criptografía, **A**dvanced **E**ncryption **S**tandard, también conocido como **Rijndael**, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos.

APS: Atención Primaria de Salud.

CICS: acrónimo en inglés de **C**ustomer **I**nformation **C**ontrol **S**ystem (en español, Sistema de Control de Información de Clientes), es un gestor transaccional, o monitor de teleproceso, que se ejecuta principalmente en mainframes IBM con los sistemas operativos z/OS o VSE. También existen versiones de CICS para otros entornos, como OS/400, OS/2, etc.

CU: casos de **u**so. (En el Proceso Unificado del Software, CU del Negocio, CU del Análisis, CU del Diseño).

DB: Una base de datos o banco de datos **(**en inglés Database.**)** es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

Fiedoms: agente que actúa a nombre de consumidor del servicio.

FTP: File **T**ransfer **P**rotocol es un protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar ficheros desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.

HTTP: el Protocolo de Transferencia de Hipertexto (**HTTP**, **H**yper**T**ext **T**ransfer **P**rotocol) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

GLOSARIO

IBM: Máquinas de Negocio Internacionales (en ingles **I**nternational **B**usiness **M**achines**)** es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática. Tiene su sede en Armonk (Estados Unidos) y está constituida como tal desde el 15 de junio de 1911, pero lleva operando desde 1888.

IT: Tecnologías de la Información (Information Technologies en sus siglas en inglés).

Java: es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible.

JCA: Java EE **C**onnector **A**rchitecture es una solución de tecnología basada en el lenguaje de programación Java para conectar servidores de aplicaciones y sistemas de información de empresa (SIE) como parte de soluciones de integración de aplicación de empresa (EAI). Mientras JDBC se usa específicamente para conectar aplicaciones Java a Bases de datos, JCA es una arquitectura más genérica para conectarse a sistemas heredados (incluyendo bases de datos).

JDBC: es el acrónimo de **J**ava **D**atabase **C**onnectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

MDA: La Arquitectura Dirigida por Modelos (**Model-Driven Architecture** o MDA) es un acercamiento al diseño de software, propuesto y esponsorizado por el Object Managemente Group. MDA se ha concebido para dar soporte a la ingeniería dirigida a modelos de los sistemas software. MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos.

MIME (**M**ultipurpose **I**nternet **M**ail **E**xtensions, Extensiones de Correo Internet Multipropósito), son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario.

GLOSARIO

MODEM: Equipo electrónico que sirve para transmitir y recibir información digital a distancia, mediante la modulación y desmodulación (en amplitud, o frecuencia o fase u otro sistema) de la señal digital.

MSF: Marco de Soluciones de Microsoft en ingles **M**icrosoft **S**olutions **F**ramework es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. (Maria A. Mendoza Sánchez Ing. Informático – UNIT, Microsoft Certified Professional – MCP, analista y desarrolladora – Team Software Perú S.A.C.

OBEs: Acrónimo de **O**bject **R**equest **B**roker. En computación distribuida es el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red. Maneja la transferencia de estructuras de datos, de manera que sean compatibles entre los dos objetos. Para ello utiliza un estándar para convertir las estructuras de datos en un flujo de bytes, conservando el orden de los bytes entre distintas arquitecturas.

OMG: el **O**bject **M**anagement **G**roup (de sus siglas en inglés Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización NO lucrativa que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para tecnologías orientadas a objetos. El grupo está formado por compañías y organizaciones de software como lo son:

- ✓ Hewlett-Packard (HP).
- ✓ IBM.
- ✓ Sun Microsystems.
- ✓ Apple Computer.

PDF: (del inglés **P**ortable **D**ocument **F**ormat, Formato de Documento Portátil) es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

Placeholders:

GLOSARIO

RDBMS: es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en inglés **R**elational **D**ata **B**ase **M**anagement **S**ystem.

RMI: (**J**ava **R**emote **M**ethod **I**nvocation) es un mecanismo ofrecido en Java para invocar un método remotamente. Al ser RMI parte estándar del entorno de ejecución Java usarlo provee un mecanismo simple en una aplicación distribuida que solamente necesita comunicar servidores codificados para Java. Si se requiere comunicarse con otras tecnologías debe usarse CORBA o SOAP en lugar de RMI.

RSA: Rational **S**oftware **A**rchitect es una herramienta integrada de Diseño y Desarrollo que proporciona un desarrollo basado en modelos con UML (Unified Modeling Language) para crear aplicaciones y servicios con una buena Arquitectura Orientada a Servicios. Soporta análisis, diseño, manejo y evolución de soluciones empresariales y de los servicios.

RUP: el Proceso Unificado del Software (en inglés **R**ational **U**nified **P**rocess.) es un proceso de desarrollo de software configurable que se adapta a proyectos que varían en tamaño y complejidad. Utiliza como lenguaje de visualización el UML (Lenguaje Unificado de Modelado en ingles Unified Modeling Language)" [M&R 1998 (Microsoft y Rational 1998 documento disponible en http://www.rational.com/uml/papers.

SOA: Arquitectura Orientada a Servicios en ingles **S**ervice **O**riented **A**rchitecture.

SOAP: (siglas de **S**imple **O**bject **A**ccess **P**rotocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

Stakeholders: Personas u organizaciones que están activamente implicadas en el negocio ya sea porque participan en él o porque sus intereses se ven afectados con los resultados del proyecto. Pueden ser los propietarios, la dirección, quienes financian, los clientes, los trabajadores, los proveedores, la competencia, la comunidad local, etc.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, **U**nified **M**odeling **L**anguage) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando

todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

URL: significa **U**niform **R**esource **L**ocator, es decir, Localizador Uniforme de Recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

WEB: World Wide Web (o la "Web") es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

WSDL: son las siglas de **W**eb **S**ervices **D**escription **L**anguage, un formato XML que constituye idioma descriptivo de los servicios Web del consorcio mundial Web.

XML: sigla en inglés de e**X**tensible **M**arkup **L**anguage (Lenguaje de Marcas Extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XP: La Programación Extrema o **E**xtreme **P**rogramming es un enfoque de la ingeniería de software formulado por Kent Beck autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). Es la más destacada de los procesos ágiles de desarrollo de software. Es una metodología factible para pequeños proyectos.