Universidad de las Ciencias Informáticas

FACULTAD 7

Aplicación del Plug-in SOA-RUP.

Caso práctico RCIE.

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

Autores: Yara De León Bridón.

Michel Rabi Cespedes.

Tutor: Ing. Luis Abel Cobo Espinosa.

La Habana, junio de 2007

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 29 días del mes de Junio del año 2007.

Autores:	Tutor:
Yara de León Bridón	Ing. Luis Abel Cobo Espinosa
_____	_____
Michel Rabi Cespedes	
_____	

DATOS DE CONTACTO

Ing. Luis Abel Cobo Espinosa, graduado en el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) en el año 1997. Acumula 10 años de experiencia laboral como profesional, durante los cuales ha aumentado sus conocimientos a través de diplomados y maestrías. Tiene amplio dominio de diversas herramientas y metodologías de trabajo que le han servido para cumplir múltiples tareas. Correo: **luiscobo@softel.cu.**

¨ Largo es el camino de la enseñanza por medio de teorías; breve y eficaz por medio de ejemplos ¨

Séneca

Agradecimientos

En primer lugar a mi familia, por inculcarme siempre el hábito de estudiar y de prepararme para la vida como profesional.

A mi novia por ayudarme y apoyarme siempre, a su familia.

A mis amigos que para no decir nombres pero que saben muy bien quienes son, les agradezco de todo corazón todo el apoyo y la fuerza que me dieron para realizar este trabajo. A mis amigos fuera de la UCI, que donde quiera que estén siempre me dieron ánimo desde que comencé en la Universidad.

Gracias a todos los profesores que me brindaron su ayuda cuando la necesite.

Gracias UCI, sin tu creación no hubiese conocido gente tan maravillosa.

Michel Rabi Cespedes.

A mi familia por depositar toda la confianza en mí durante tantos años de estudio

Gracias a mis padres, especialmente a mi mamá por estar siempre a mi lado, por ser compañera, amiga, por tanto sacrificio, por entenderme, por existir, gracias

A mis amigos dentro y fuera de la UCI que tanto me ayudaron y soportaron en todo este tiempo y que han influido tanto en mi formación personal como profesional, especialmente a Ibet, Anaima, Dulce, Yanelys, Maylen, Mara, Yeni, Lily, Yaritza, Karelis, Dayanis, Roque, a mis negros Ariel y Victor.

A todos los profesores y profesionales me han apoyado en estos cinco años.

Gracias a la gran obra que es la Universidad de Ciencias Informáticas, por brindarme todos los medios necesarios para mi formación

A todos que de una forma u otra han hecho huellas en mi vida

Gracias

Yara de León Bridón

Dedico este trabajo y todos los méritos alcanzados hasta este momento en mi vida,

A mi madre y a mi abuela Enma Luz.

Michel Rabi Cespedes.

A mi mamá por tanto amor y comprensión

A mi papá, por demostrarme que mientras haya vida hay esperanzas

A mi abuela por la paciencia

Yara de león Bridón.

RESUMEN

El presente trabajo se desarrolló con el objetivo de conocer la guía que propone el Plug-in SOA-RUP y con esto efectuar los cambios necesarios en el modelamiento de la ingeniería de software del Registro del Codificador Internacional de Enfermedades (RCIE), teniendo en cuenta la arquitectura orientada a servicios en que fue desarrollado. La investigación surge por la necesidad de establecer un documento rector en el uso de esta arquitectura en el proyecto APS, y específicamente en el módulo RCIE. Además, permitirá obtener un Modelo de Servicios, que corresponda con lo que se ha implementado hasta este momento.

Se ha considerado lo antes modelado en el Rational Rose Enterprise Edition del 2003 para analizarlo y determinar, a partir de ahí, el camino a seguir. La principal herramienta utilizada, es el Rational Unified Process Builder; que publica la guía propuesta por el Plug-in, donde aparecen los pasos fundamentales para desarrollar una aplicación basada en SOA, y específicamente un Modelo de Servicios. Otra herramienta es Rational Software Architect (RSA), recomendada por el Plug-in, pues incluye nuevas actividades, artefactos y estereotipos para la realización de los modelos que se proponen.

Con la elaboración de esta investigación se obtienen una serie de beneficios claves para el desarrollo de RCIE y del proyecto APS de forma general. Esta serviría como ejemplo y plantilla para la realización de un Modelo de Servicios SOA con metodología RUP, y por consiguiente, se podría convertir en un material de consulta en el proyecto, y puede ser extensible hasta cualquier nivel en el proceso de informatización del Sistema Nacional de Salud (SNS).

PALABRAS CLAVES

Registro de Codificación Internacional de Enfermedades (RCIE).

Arquitectura Orientada a Servicios (SOA).

Plug-in SOA-RUP.

Proceso Unificado del Software (RUP).

Modelo de Servicios.

Servicios.

TABLA DE CONTENIDOS

INTRODUCCIÓN

INTRODUCCIÓN

El Sistema Nacional de Salud (**SNS**) de Cuba; universal, gratuito y al alcance de todos los cubanos sin distinción de raza, procedencia social o religión, se ha desarrollado a partir de un concepto social de la salud que en el momento actual es preciso decir que rebasa los límites del individuo y que abarca su relación e interacción con el medio donde éste se desarrolla. [1]

La Revolución, conjuntamente con el Ministerio de Salud Pública (**MINSAP**), ha trazado grandes estrategias para la reorientación del SNS hacia la Atención Primaria (**APS**), fundamentalmente hacia los Médicos y Enfermeras de la Familia, dúo que conforma el Equipo Básico de Salud (**EBS**), protagonistas de la Atención Primaria de Salud. [2]

La informatización del Sistema Nacional de Salud en Cuba tiene como objetivos desde el año 2000, acercar eficientemente y con calidad, la prestación de los servicios de salud a la población a través del Programa General de Informatización. Para llevar a cabo el Proceso de Informatización del SNS abarcando a la APS y al policlínico como eje fundamental, fue encomendada esta tarea por la dirección del MINSAP y el Ministerio de Informática y Comunicaciones (**MIC**), a la Empresa SOFTEL, empresa cubana dedicada a la ejecución de soluciones informáticas para la salud, que en conjunto con la Universidad de las Ciencias Informáticas (**UCI**) y Médicos Especialistas en Medicina General Integral en calidad de expertos funcionales, tienen la misión en el marco del Proyecto APS de elaborar un producto de software que facilite la gestión de la información y la toma de decisiones en este nivel de atención. [3]

El SNS está organizado por los niveles de dirección Nacional, Provincial y Municipal y según la complejidad de las acciones preventivas, curativas y de rehabilitación, así como el grado de especialización de los servicios, en los niveles de Atención Primaria, Atención Secundaria y Atención Terciaria de Salud.

El proyecto APS, desarrollado en la Universidad de las Ciencias Informáticas, por parte de los estudiantes de la Facultad 7 vinculada al segundo perfil de Salud, junto a la empresa SOFTEL, cuenta con varios módulos a informatizar dentro de los cuales se destacan: Registro de Población (**RPOB**), Registro de Enfermedades de Diagnosticación Obligatoria (**REDO**), Registro de Problemas de Salud de la Atención Primaria (**RPSAP**), Registro del Codificador Internacional de Enfermedades (**RCIE**), entre otros, siendo este último uno de los que ha llevado la delantera en este proceso de informatización.

El objetivo principal de RCIE, es clasificar las enfermedades y problemas relacionados con la salud de los pacientes atendidos en dicho nivel en un lenguaje unificado y estandarizado con el fin de disponer de información apropiada para establecer prioridades así como para el planeamiento y evaluación de los servicios. La adopción de esta clasificación en la gestión en la APS traerá consigo beneficios asistenciales tales como: [4]

✓ Originará un salto de calidad para el diagnóstico y la precisión de las enfermedades, causas de muerte u otros trastornos o condiciones de salud.

✓ Permitirá, junto con la consiguiente mejor preparación del personal de la salud cubano, una mejor satisfacción de la necesidad del paciente de tener un diagnóstico o una respuesta a sus interrogantes de lo que le sucede, ya se trate de problemas de personas que se sienten enfermas, o de problemas planteados por personas que se consideran sanas pero que buscan evaluación, atención y consejo experto.

✓ Brindará servicios a los restantes registros que contiene la APS para su desarrollo en la gestión de la información. [5]

Para llevar a cabo la ingeniería de software de este producto y del resto de los módulos del proyecto, se ha utilizado como metodología el **RUP** (Proceso Unificado de Desarrollo), utilizando la herramienta **Rational Rose Enterprise Edition** en su versión del 2003 debido a todas las funcionalidades que brinda en el desarrollo del ciclo de vida de un software. En esta versión no existe una configuración para realizar un **modelo de servicios** basado en arquitectura orientada a servicios más conocida como **arquitectura SOA**.

A pesar de que existe esta situación con la herramienta, y debido a los servicios de información que RCIE brinda a los restantes registros, se ha diseñado bajo la óptica del uso de la arquitectura SOA cumpliendo además con la política establecida por el MINSAP.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación. En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la

utilización de **Servicios Web** (empleando **SOAP** y **WSDL**) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implantación.

Recientemente la Empresa de software IBM, lanzó al mercado el **Plug-in SOA-RUP** (Ver Anexo 1), el cual esta diseñado para brindar una metodología de trabajo para modelar servicios en diagramas UML. El mismo constituye una actualización para el Rational Unified Process, en él se introduce una guía para el arquitecto y el diseñador del software para desarrollar un modelo de servicio, el cual está representado por un conjunto de artefactos de servicios que se utilizan como entrada para las actividades en las fases de implementación y prueba:

- ✓ Mensaje.
- ✓ Servicio.
- ✓ Canal de Servicio.
- ✓ Colaboración de Servicio.
- ✓ Entrada de Servicio.
- ✓ Partición de Servicio.
- ✓ Proveedor de Servicio.
- ✓ Consumidor de Servicio.
- ✓ Especificación de Servicio.

Debido a las dificultades antes mencionadas, surge la siguiente **Situación Problémica:**

El módulo RCIE culminó su implementación y el proceso de Ingeniería de Software, este último fue realizado en la herramienta Rational Rose en su versión del 2003 y en la misma no existe una configuración para modelar arquitectura orientada a servicios.

Como no se tenía el conocimiento de una metodología específica para aplicaciones basadas en SOA, hasta el momento, se han realizado diagramas provisionales en los módulos que consumen servicios de RCIE.

Esta situación se produjo al no existir una documentación sobre arquitectura SOA dirigida a RCIE, y por lo tanto no hay una relación traceable entre la implementación y lo modelado.

Considerando lo anteriormente expuesto y la necesidad de buscar una respuesta a estas dificultades, que frenan el desarrollo eficiente del módulo RCIE y del proyecto APS de forma general, los esfuerzos del presente trabajo estarán centrados en resolver el siguiente **problema**:

Problema Científico

¿Cómo realizar el análisis y diseño con Arquitectura Orientada a Servicio (SOA) en el módulo RCIE?

El **Objeto de la Investigación** se centra en el:

Proceso de ingeniería de software con arquitectura SOA utilizando RUP.

El **Campo de Acción** es:

Proceso de ingeniería de software con arquitectura SOA para el módulo RCIE en el proyecto APS.

El **Objetivo General** que se persigue es:

Realizar los cambios necesarios en la ingeniería del software del módulo RCIE basándose en el Plug-in SOA-RUP implementado para el Rational Unified Process Builder (Ver Anexo 2), de forma tal que se adapte a una arquitectura orientada a servicios.

Para el cumplimiento al objetivo planteado, se proponen las siguientes **Tareas**:

- ✓ Valorar las tendencias de las diferentes empresas en Cuba y en el mundo que utilizan SOA.
- ✓ Examinar el uso del Plug-in SOA-RUP para el Rational.
- ✓ Indagar en el análisis y diseño realizado para el módulo RCIE hasta el momento.
- ✓ Formalizar las especificidades de RCIE con el Plug-in SOA-RUP.

Para realizar una descripción detallada, este documento muestra el resultado de la investigación de la siguiente forma:

CAPÍTULO 1: Contiene la fundamentación teórica del tema tratado en la investigación. Se describen los conceptos fundamentales asociados al dominio del problema, y también se puede encontrar una descripción de las tecnologías utilizadas.

CAPÍTULO 2: Se describen las características del sistema actual de RCIE y se exponen las soluciones que brinda el Plug-in SOA-RUP para utilizar esta metodología.

CAPÍTULO 3: Se realizan los modelos de servicios de RCIE que lo relacionan con la gestión de la información de otros registros del proyecto APS.

El documento cuenta además con las conclusiones del trabajo, algunas recomendaciones a tener en cuenta en la continuidad del proyecto, las referencias bibliográficas y un conjunto de anexos que permitirán una mejor ilustración del estudio realizado.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórico conceptual para el desarrollo de este trabajo. Se exponen además los antecedentes e intentos de integración entre sistemas informáticos en el área de la salud pública específicamente en el caso práctico del módulo RCIE del proyecto APS, tomando esto como punto de partida, para la concepción de la solución en cuestión. Además se realizará un análisis de las tecnologías y metodologías sobre las cuales se llevará a cabo el presente trabajo.

1.1 Sistema de Información para la Salud (SISalud)

Con el objetivo de satisfacer las necesidades en todos los niveles de salud, y como parte del apoyo a la automatización del SNS, se está desarrollando un sistema complejo que permite la integración de componentes, servicios y aplicaciones relacionados con dicho programa, cuyo nombre es Sistema de Informatización para la Salud (**SISalud**).

SISalud estará compuesto por el Registro Informatizado de Salud (**RIS**), el Sistema Automatizado de Atención Primaria (**SIAP**), el Sistema Informatizado de Gestión Hospitalaria (**SIGH**) y el Sistema Informatizado de Atención Especializada (**SIAE**). De ellos el sistema que centra la atención del presente trabajo es el RIS.

El RIS está formado por los registros que serán administrados o gestionados a nivel nacional y que integran el Registro No Médico Informatizado de Salud (**RNMIS**), dentro del cual se encuentra el Registro de Clasificación Internacional de Enfermedades (RCIE), y por los registros que pueden ser accesados desde cualquier nivel de atención o institución del salud para ganar continuidad en el seguimiento del paciente, agrupándose estos en el Registro Médico Informatizado de Salud (**RMIS**). [6]

1.2 Registro de la Clasificación Internacional de Enfermedades (RCIE).

Con el proyecto RCIE se propone informatizar el clasificador de igual nombre propuesto por la Organización Mundial de la Salud (OMS) en su Décima Revisión (CIE-10), que dispone el uso de una codificación alfanumérica y un marco conceptual para la descripción precisa de las enfermedades y problemas de salud, apoyándose en un lenguaje unificado y estandarizado. [7]

El módulo RCIE tiene dos objetivos o propósitos bien definidos:

✓ El primero y fundamental es brindar información acerca de los problemas de salud, en primer lugar a cualquier persona interesada en el tema, principalmente al personal médico, paramédico y directivos de una Unidad de Salud de forma que la misma sea oportuna y útil para la toma de decisiones.

✓ En segundo lugar, facilita los cambios y actualizaciones de la información, necesarios en el Registro Internacional de Enfermedades y Problemas Relacionados con la Salud por parte del personal pertinente. [8]

1.3 Estructura Jerárquica de RCIE

Este registro está estructurado jerárquicamente desde un nivel superior hasta el inferior de la siguiente forma:

- ✓ **Capítulo**

Corresponde al primer nivel de la estructura jerárquica del codificador. El primer carácter del código de cada capítulo se asocia con una letra y cada letra se asocia con un capítulo en particular, con sus excepciones. [9]

- ✓ **Grupo Segundo Nivel**

Corresponde al segundo nivel de la estructura jerárquica del codificador. Es la división en bloques o grupos homogéneos de categorías de tres caracteres dentro de un capítulo. [10]

- ✓ **Categoría**

Corresponde al tercer nivel de la estructura jerárquica del codificador. Dentro de cada grupo, algunas de las categorías de tres caracteres corresponden a afecciones únicas, seleccionadas según su frecuencia, gravedad o vulnerabilidad a las acciones de salud pública. [11]

- ✓ **SubCategoria**

Corresponde al cuarto nivel de la estructura jerárquica del codificador. La mayoría de las categorías de tres caracteres están subdivididas por medio de un carácter numérico, permitiendo hasta 10 subcategorías. [12]

- ✓ **Problemas de Enfermedad**

Se está en presencia de un problema de Salud o enfermedad cuando un paciente acude en busca de atención médica en cualquier nivel de atención medica. [13]

1.4 Historia del arte del Plug-In SOA-RUP.

El Plug-In SOA RUP forma parte de una de las guías que propone el Rational Unifield Process Builder que conforma la Suite del Rational. El mismo es una metodología que propone artefactos, fases, trabajadores y herramientas que se deben asumir para el desarrollo y modelado de una aplicación basada en RUP y que utilice arquitectura SOA.

Este se ha tenido en cuenta para la elaboración del presente trabajo debido a que el modelado de la arquitectura del módulo RCIE está basado en RUP por las facilidades y ventajas que esta metodología propone para el desarrollo y liberación organizada y satisfactoria de un producto.

1.4.1 Antecedentes en la UCI.

El Plug-In SOA-RUP es bastante reciente por lo que aún en la universidad no se ha desarrollado un sistema que lo utilice, pero si se han implementado sistemas basados en SOA como es el caso del proyecto Intranet 2 que se encuentra elaborándose en la Facultad 10. Este es un proyecto encaminado a eliminar todos los posibles errores o inconvenientes que presenta la Intranet actual en la Universidad, y por tanto tiene como objetivo brindar servicios de gran estabilidad, rapidez y robustez, y que además pueda interactuar con otros sistemas, pero en la actualidad no se ha profundizado en un modelamiento de ingeniería de software conveniente para ello.

Otro proyecto también de la Facultad 10 es el Visión Portal en contrato con la compañía petrolera de Venezuela PDVSA. El mismo constituye una nueva Intranet de Servicios para esta compañía y está basado en SOA. Para el modelado de la arquitectura se utilizó como herramienta Case el Visual Paradigm pero no se llevó a cabo una metodología RUP, pues esta está más centrada en casos de uso y la que se puso en práctica y que respondía a sus necesidades estaba orientada fundamentalmente a procesos y recursos de procesos.

Es por ello que se decidió por el uso de un acercamiento conveniente con modelos de arquitectura planteados por los líderes de la industria de SOA: IBM y Software AG.

Las aplicaciones vinculadas fundamentalmente a la informatización del SNS, presentan más profundidad en el tema, como es el ejemplo práctico de RCIE, perteneciente al proyecto APS.

¿Que solución se ha dado para la construcción y proceso de ingeniería de software de una arquitectura orientada a servicios dentro del proyecto APS, y en el caso específico del módulo RCIE (sin el uso del Plug-in SOA-RUP) hasta la actualidad?

Para la construcción de una arquitectura SOA se han tenido los elementos más importantes que ella plantea:

- ✓ **Consumidores de Servicios**
- ✓ **Proveedores de Servicios**
- ✓ **Bus Empresarial de Servicios**

Este módulo está concebido completamente sobre arquitectura orientada a servicios y su desarrollo basado en componente, empleando Servicios Web XML, específicamente SOAP. Utiliza **PLASER** (PLAtaforma de SERvicios) como **Bus Empresarial de Servicios** para facilitar la programación y homogeneidad de los componentes.

Inicialmente la arquitectura SOA que se utiliza propone el uso del lenguaje XSL, la misma usa el XPATH como compilador y XSLT como encargado de transformar los ficheros XML mediante un método de la clase Fachada. Finalmente los ficheros con extensión XSL muestran al usuario una respuesta con el servicio o información solicitada.

Cada módulo tiene una clase Fachada que hereda de la clase Fachada abstracta la cual sirve como interfaz entre los módulos y PlaSer, debido a que la misma está compuesta por objetos de tipo PLASER_xml, PLASER_client y PLASER_xslt.

La solución que se ha dado para representar este proceso en el flujo de análisis y diseño para cada caso de uso de RCIE es mediante clases interfaces y controladoras "**ficticias**" para encapsular los ficheros que tengan que ver con el diseño y las funcionalidades del módulo.

Esto quiere decir que por ejemplo en la capa de presentación se han diseñado clases (tanto interfaces como controladoras), en los diagramas de clases del diseño (y por consiguiente en los diagramas de secuencias pertenecientes a cada caso de uso), que en realidad no existen, para de esta forma, encapsular métodos que cumplen con la funcionalidad de esa supuesta clase.

En la capa de negocio de lo que se trata es de la existencia de una clase abstracta que acoge los métodos del negocio, los cuales fueron programados indiferentemente para adicionarlo a esta clase (**Confplaser_server**).

La transferencia de servicios entre RCIE y demás componentes contienen todas estás características. La solución que se le ha dado en el proceso de flujo de análisis y diseño para el caso de los consumidores de RCIE, que están en fase de desarrollo o en su proceso de culminación, es la siguiente:

En RPSAP:

✓ En el modelo de Casos de Uso del Sistema se representó a RCIE como un actor relacionado con los casos de uso específicos que consumen servicios, en este caso son Gestionar Discapacidades y Gestionar Enfermedades de RPSAP (Ver Anexo 3).

✓ En los diagramas de realización de estos casos de usos del sistema de las clases del análisis se representó a RCIE como una clase estereotipada como servicio (Ver Anexos 4 y 5).

En REDO:

✓ En el modelo de Casos de Uso del Sistema se representó a RCIE como un actor relacionado con los casos de uso específicos que tienen que ver con el consumo de servicios, en este caso es Gestionar Enfermedades de REDO (Ver Anexo 6).

1.4.2 Actualidad Internacional del Plug-in SOA-RUP

En la esfera internacional las investigaciones realizadas no han aportado una basta experiencia con respecto al uso de este Plug-in. Sin embargo se puede esclarecer que en el presente el Plug-in SOA-RUP ha sido actualizado y ya se encuentra su tercera versión 2.4 debido a que se han encontrado algunas dificultades en el momento de modelar proyectos basados en SOA:

✓ Falta de documentación para los requisitos del software para SOA (no ayudan los del Plug-in SOA-RUP).

✓ Cuál es el intercambio entre lo que realmente necesita el proyecto y lo que será bueno tener en cuenta para reutilizar, debido a que se supone que los servicios deben ser reutilizados.

✓ Los casos de usos para un servicio tienen más funciones que un caso de uso.

- ✓ Falta de conocimiento del legado del sistema y ningún acceso a él.
- ✓ La ausencia de un modelo estructurado para las entidades del negocio devueltas por un servicio, es decir, por ejemplo la información de un cliente puede dividirse en dos: información de contacto por un lado e información del producto por el otro, lo que puede generar dos casos de usos: obtener información de contacto y obtener información de producto. [14]

Como respuesta y solución a estas deficiencias, se creó la versión 2.4. Esta es la tercera actualización al Proceso Unificado de Desarrollo (RUP) enfocada sobre la Arquitectura Orientada a Servicios (SOA). La misma representa un hito mayor en la guía de RUP alrededor de SOA así como provee un método unificado que combina el contenido de SOA-RUP con el contenido de IBM para Servicios de Negocio Globales (GBS) y el método de la Arquitectura y Modelamiento Orientados a Servicios (SOMA). EL método SOMA ha sido usado satisfactoriamente por IBM en un número de clientes comprometedor, y mientras este fue inicialmente desarrollado servía como palanca al Método Global de Servicios de IBM ya existente (el Método de GS), era sentido que en el área de SOA, IBM y los clientes podrían beneficiarse más de un acercamiento a un método unificado aprovechando la tenencia de los dos métodos separados. Sin embargo las versiones antiguas del Plug-in constituyen para la elaboración de este trabajo una buena guía según los objetivos trazados. [15]

1.5 Tendencias y tecnologías actuales a considerar.

En este epígrafe se aborda acerca de los conceptos fundamentales relacionados con la arquitectura, tecnologías, herramientas y metodologías necesarias en el entendimiento del proceso de modelado de RCIE utilizando el Plug-in SOA-RUP.

1.5.1 Modelo Cliente Servidor.

La arquitectura Cliente-Servidor, es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificar las actualizaciones y mantenimiento del sistema. En una arquitectura monolítica no hay distribución; los tres niveles tienen lugar en el mismo equipo. En el modelo cliente-servidor, en cambio, el trabajo se reparte entre dos ordenadores. [16]

Se puede decir que todas las aplicaciones tienen la misma arquitectura básica y se pueden subdividir en tres partes:

- ✓ **Interfaz de Usuario:** La presentación del usuario, con las entradas de dato y las pantallas de consultas.
- ✓ **Reglas de Negocio:** El Procesamiento de la información.
- ✓ **Accesos a Dato:** El control del almacén de datos. [17]

1.5.2 Arquitectura de 3 Capas

La arquitectura de tres niveles es la generalización de la arquitectura cliente - servidor donde la carga se divide entres partes con un reparto claro de funciones: una capa de presentación, otra parte para las reglas lógicas del negocio, denominada capa de negocio y la capa de datos para el almacenamiento de los mismos. Es decir, está soportado sobre un nivel de abstracción creciente, lo cual permite a los desarrolladores la fragmentación de un problema complejo en una secuencia de pasos incrementales. También proporciona una amplia reutilización, pues los datos abstractos pueden ser utilizados por diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces. [18]

Las capas de presentación (clientes), de negocio y datos pueden residir en un único servidor, aunque lo más común es que haya una multitud de servidores donde resida este modelo de arquitectura en dependencia de la complejidad de estas capas. Por ejemplo las capas de negocio y datos pueden coexistir simultáneamente en un mismo servidor, ahora se deber tener en cuenta un grupo de elementos que de presentarse, obligarían a realizar la separación de estas, uno de ellos está dado por el volumen y tamaño de la base de datos, considerándose que la misma pueda seguir creciendo en el tiempo una vez generalizado el sistema informático. Por el contrario si la complejidad fuese en el negocio este es subdividido, realizando peticiones sobre una misma base de datos. Este modelo es el que se encuentra implementado para el Registro Informatizado de Salud y será generalizado durante el proceso de desarrollo del Sistema de Información para la Salud. [19]

1.5.3 SOA (Arquitectura Orientada a Servicios).

SOA “Arquitectura Orientada a Servicios” es un estilo de arquitectura cuyo propósito primordial es lograr un débil acoplamiento entre los componentes de software que interactúan entre si. Bajo este modelo se entiende cada servicio como un componente atómico que realiza una unidad de trabajo para realizar una tarea particular. [20]

Sin embargo, SOA es más que una arquitectura de software, pues ha dado paso a todo un modelo tecnológico que considera elementos tales como Motores de Procesos del Negocio, Monitoreo de la Actividad, etc. [21]

1.5.3.1 SOA como Tendencia.

SOA no es un concepto nuevo, desde hace más de 30 años la industria viene planteándose la necesidad de crear arquitectura de software débilmente acopladas. Acoplamiento débil significa que el cliente de un servicio es esencialmente independiente de la construcción de ese servicio. [22]

Las Arquitecturas Orientadas a Servicios están motivadas por la creciente necesidad de los negocios de responder con rapidez a los cambios en el entorno comercial en que se desenvuelven. Esto los lleva a tener que cambiar sus sistemas tecnológicos con esa misma rapidez y para lograrlo es necesario que los componentes de esta infraestructura sean tan reutilizables y poco interdependientes que permitan una rápida reestructuración de los mismos. [23]

1.5.3.2 Elementos básicos que conforman una SOA.

- ✓ **Proveedores de Servicios.**
- ✓ **Consumidores de Servicios.**
- ✓ **Bus Empresarial de Servicios.**

En realidad todo componente dentro de una organización puede ser tanto proveedor como consumidor de servicios. Todos los servicios interactúan entre si a través de un Bus Empresarial de Servicios (o ESB por sus siglas en Inglés). [24]

Además podemos definir otros elementos participantes dentro de una arquitectura SOA tales como:

- ✓ **Cliente – Proveedor.**

Se entiende Cliente como el componente que invoca un servicio provisto por un proveedor.

- ✓ **Concepto de servicio.**

Un servicio es una unidad de trabajo realizada por un componente de software a fin de conseguir un resultado específico. El servicio debe ser alcanzable por parte de los consumidores a través de una interfaz programática.

- ✓ **Utilización de Web Services.**

La arquitectura orientada a servicios no especifica necesariamente que deben ser brindados a través de un protocolo específico. Los Web Services son en realidad un conjunto de estándares que definen un protocolo de invocación remota de servicios, basados en HTML y XML. Si bien, son también un mecanismo adecuado y en muchos casos recomendable para implementar servicios, no son el único. Es importante que estas arquitecturas soporten múltiples protocolos a fin de cumplir al máximo su visión de brindar un modelo de integración para toda la plataforma tecnológica.

- ✓ **ESB.**

El Enterprise Services Bus o ESB (Bus Empresarial de Servicios) es muy probablemente el componente más importante. El Enterprise Services Bus se basa en la mejor práctica en patrones de diseño para integración de aplicaciones, típicamente conocido como **HUB & SPOKE.** [25]

Este patrón plantea la existencia de un componente de mediación que provee servicios de ruteo, transformación de mensajes, publicación y distribución de eventos y soporte para múltiples protocolos. [26]

1.5.3.3 ¿Qué es un servicio en SOA?

Es una función de aplicación empaquetada como un componente reutilizable para ser usado en un proceso de negocio. Proporciona información o facilita el cambio de datos de negocio de un estado válido y consistente a otro. Es autocontenido y sin estado. La implementación concreta de un servicio SOA no es importante. A través de protocolos de comunicación bien definidos, pueden ser invocados de manera que se hace hincapié en la interoperabilidad y en la transparencia de localización. [27]

1.5.3.4 Protocolos utilizados en SOA.

✓ **Web Services:** que en realidad se refieren a un conjunto de protocolos, particularmente, **WSDL** (Web Services Definition Language), XML y SOAP (Simple Object Access Protocol). Existen además un conjunto amplio de protocolos que se sientan sobre estos para brindar características ampliadas de seguridad, confiabilidad y administración a los Web Services.

✓ **UDDI:** (Universal Description Discovery and Integration) que define el protocolo para describir, encontrar e integrar servicios. [28]

1.5.3.5 Mensajes en un ambiente SOA

El mensaje es el contenido de la invocación del servicio que lleva la información propia del negocio necesaria para la realización de las operaciones atadas al mismo.

✓ **Concepto y uso de interfaces.**

Un aspecto importante dentro de la Arquitectura Orientada a Servicios es mantener una adecuada separación entre la implementación de un servicio y su interfaz. La interfaz del servicio define la forma en que este puede ser invocado a través preferiblemente de un protocolo estándar como Web Services.

✓ **Relación Interfaz – Componente – Mensaje.**

El componente brinda una funcionalidad (servicio) a través de una interfaz en la cual se intercambia un mensaje que contiene la información de negocio necesaria para la realización de la tarea específica.

✓ **La infraestructura de mensajería dentro de la arquitectura.**

La infraestructura de mensajería es el canal preferido de comunicación entre servicios, principalmente en implementaciones complejas de la arquitectura donde características como confiabilidad, seguridad y aseguramiento de la entrega se vuelven críticos para el éxito. [29]

1.5.3.6 ¿Por qué usar SOA?

Existen varias razones para que una empresa adopte un enfoque SOA, y más concretamente un enfoque SOA basado en Web Services:

✓ **Reutilización**.

El factor fundamental en el cambio a SOA es la reutilización de los servicios de negocio. Las funciones de negocio, dentro de una empresa y con los socios del negocio, pueden ser expuestos como Web Services y ser reutilizadas para cubrir nuevas necesidades de negocio.

- ✓ **Interoperabilidad.**

El objetivo de una arquitectura débilmente acoplada es que los clientes y servicios se comuniquen independientemente de la plataforma en que residan. Los protocolos de comunicación con Web Services son independientes de la plataforma, lenguaje de codificación y sistema operativo por lo que facilitan la comunicación con los socios del negocio.

- ✓ **Escalabilidad.**

Como los servicios de SOA están débilmente acoplados, las aplicaciones que usan esos servicios escalan fácilmente. Esto es debido a que existe muy poca dependencia entre las aplicaciones clientes y los servicios que usan.

- ✓ **Flexibilidad.**

Es otra de las características que proporciona el acoplamiento débil entre los servicios. Cualquier cambio en la implementación de uno de ellos no afectaría al resto siempre que se mantenga la interfaz.

- ✓ **Eficiencia de coste.**

Las arquitecturas SOA se basan en la exposición de servicios ya existentes para ser reutilizados. Al usar Web Services para exponer estos servicios, se reutilizan la infraestructura Web existente en virtualmente todas las organizaciones por lo que se limita considerablemente el coste. [30]

1.5.4 CBA (Arquitectura Basada en Componentes).

La Arquitectura Basada en Componentes tiene como objetivo construir aplicaciones complejas mediante ensamblado de módulos (**componentes**), que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones. Cada componente debe describir de forma completa las interfaces que ofrece, así como las interfaces que requiere para su operación. Y debe operar correctamente con independencia de los mecanismos internos que utilice para soportar la funcionalidad de la interfaz. Es actualmente una de las más prometedoras técnicas para incrementar la calidad del

software, abreviar los tiempos de acceso al mercado y gestionar el continuo incremento de su complejidad. [31]

Ventajas de desarrollar software basado en componentes:

- ✓ **Reutilización del software:** Lleva a alcanzar un mayor nivel de reutilización de software.
- ✓ **Simplifica las pruebas:** Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- ✓ **Simplifica el mantenimiento del sistema:** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- ✓ **Mayor calidad:** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. [32]

1.5.5 RPC (Remote Procedure Call). Llamada a Procedimiento Remoto

RPC es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo fue propuesto inicialmente por Sun Microsystems como un gran avance sobre los socket usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC. [33]

1.5.6 Tecnología XML- Web Service.

Es una unidad lógica de aplicación que incluye datos y servicios y permite la comunicación entre sistemas heterogéneos, usando el paradigma de RPC, codificando los mensajes en XML y transportándolos por HTTP. Las aplicaciones obtienen acceso a los Servicios Web XML mediante protocolos Web y formatos universales de datos como HTTP, XML y SOAP. Este concepto surgió por naturaleza propia del mundo XML, Cómputo distribuido y el Web, por lo que no tiene un padre creador; se define desde el punto de vista funcional. [34]

1.5.7 Servicio Web XML

Es una unidad programable a la que sistemas muy dispares pueden tener acceso a través de Internet. Estos servicios dependen fundamentalmente de la aceptación generalizada de XML, HTTP y otros estándares de Internet que admiten la interoperabilidad.

Un servicio Web XML se puede utilizar internamente por una sola aplicación, o bien exponerse de forma externa en Internet para que puedan usarlo varias aplicaciones. Estos servicios son accesibles a través de una interfaz estándar, lo que permite que sistemas heterogéneos puedan trabajar en común como una sola red de informática. [35]

1.5.8 Alta Cohesión y Bajo Acoplamiento.

La alta cohesión significa que la información que gestione un servicio determinado, debe ser coherente y estar en la mayor medida de lo posible relacionada con la información proporcionada por este. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeña por el resto de los elementos. Un ejemplo de baja cohesión son servicios que realizan demasiadas tareas. En todas las metodologías de desarrollo se considera la factorización, proceso que permite la creación de los denominados paquetes de servicio. El bajo acoplamiento es la idea de tener las clases y los servicios lo menos ligados entre si, de forma tal que en caso de producirse una modificación en alguno de ellos, se tenga la mínima repercusión posible en el resto de los servicios que conforman el componente, potenciándose la reutilización y disminuyendo la interdependencia. [36]

1.5.9 Middleware.

Es una capa de software intermedio entre el cliente y el servidor. Es la capa de software que permite gestionar los mecanismos de comunicaciones.

Es un conjunto de interfaces y protocolos estándares de comunicación. Con interfaces estándares de programación, es fácil de implementar una misma aplicación en una variedad de tipos de servidores y de puestos de trabajo. Esta tiene un beneficio para los clientes puesto que estos compran aplicaciones no servidores, los clientes solo elegirán entre aquellos servidores donde se ejecuten las aplicaciones que ellos deseen.

Se necesitarán protocolos estándares para enlazar las distintas interfaces de servidor con los clientes que necesiten acceder a ellos. Existe una gran variedad de paquetes middleware, simples o complejos. Todos

tienen en común la capacidad de ocultar las complejidades y diferencias de los diferentes protocolos de red y sistemas operativos.

La finalidad básica del middleware es hacer que una aplicación o usuario del cliente acceda a una serie de servicios del servidor sin preocuparse de las diferencias entre servidores. [37]

1.5.10 Plataforma PlaSer.

La plataforma de Servicios (PLASER), está conformada fundamentalmente por varias clases en PHP, es una librería que puede o no ser usada para que un componente se integre al RIS, pero de no ser usada la seguridad corre por parte del programador. En esta versión, PLASER solo soporta como llamada RPC el protocolo SOAP, pero en futuras versiones se piensa en otros protocolos de transportes o incluso el acceso local a código a nivel de File System, de forma tal que para el programador es totalmente transparente si la invocación del proceso es remoto, local, por SOAP, directamente a código, etc. PLASER puede verse como una especie de **framework** sobre el cual se instalan los módulos o componentes del sistema. En su estructura encapsula varios componentes:

- ✓ **Pear SOAP**: Biblioteca para la comunicación SOAP.
- ✓ **DBX**: Extensión para la abstracción del acceso a datos.
- ✓ **XSL:** Extensión para transformaciones de ficheros XML**.**

Este sistema esta concebido completamente sobre Arquitectura Basada en Componentes y Orientada a Servicios, usando el paradigma de XML Web Services específicamente SOAP. En su concepción se han utilizado estándares actuales y normas abiertas. PLASER constituye una plataforma sobre la que se pueden desplegar aplicaciones XML – Web Services, además facilita la programación y homogeneidad de los componentes. PLASER desde el punto de vista estructural permite trabajar con cualquier base de datos que cumpla con la norma SQL ANSI 92; pero desde el punto de vista de implementación solo trabaja con las bases de datos soportadas por el componente DBX, ya que encapsula a dicho componente y lo utiliza para el acceso a bases de datos. [38]

1.5.11 XML (Extensible Markup Language).

Es un meta-lenguaje que permite definir lenguajes de marcado adecuados a usos determinados. En la práctica corresponde a un estándar que permite a diferentes aplicaciones interactuar con facilidad a través de La Red. XML fue creado al amparo del World Wide Web Consortium (W3C) organismo que vela por el

desarrollo de WWW partiendo de las amplias especificaciones de SGML. Su desarrollo se comenzó en 1996 y la primera versión salió a la luz el 10 de febrero de 1998. La primera definición que apareció fue: Sistema para definir, validar y compartir formatos de documentos en la Web. [39]

¿Por qué XML?

- ✓ Está universalmente aceptado para transmitir datos en sistemas distribuidos.
- ✓ Es una especificación oficial W3C desde Feb. 1998.
- ✓ Es un metalenguaje de descripción de datos. Permite la autodescripción.
- ✓ Importante: la mayoría de los Sistemas Operativos vienen con un parser XML.
- ✓ Parser lee el documento XML y lo expone al programa cliente.
- ✓ Web Service usa XML para describir la información que se pasa en la invocación, como por ejemplo, parámetros.
- ✓ Otros protocolos, que se usan en Web Services, se basan en XML.
- ✓ Un documento XML tiene elementos delimitados por tags. [40]

1.5.12 XHTML (Xtensible Hypertext Markup Language).

HTML carece de grandes cantidades de marcas semánticas por lo que dificulta la descripción y el intercambio de datos y la mayoría de los documentos especificados en este lenguaje, en la actualidad, están plegados de código innecesario lo que hace que la velocidad de descarga aumente considerablemente. Desde finales de 1998, ha habido una migración hacia tecnologías relacionadas con XML. Su éxito no fue una sorpresa, pues este lenguaje permite a los desarrolladores la capacidad de crear un vocabulario personalizado, a la vez que separa la presentación de la estructura, lo que hace que el mantenimiento y el análisis gramatical sea más fácil. [41]

XHTML es el acrónimo de eXtensible Hypertext Markup Language (Lenguaje de marcado de hipertexto extensible). No es más que una reformulación de HTML 4.01 de acuerdo a las reglas de XML 1.0, en realidad está diseñado para que HTML vuelva a sus raíces y lo convierta en un grupo de marcación que describa una clase en particular de contenido, sin importar cómo se visualizará este. Debido a que está especificado de acuerdo a XML, puede beneficiarse de muchas tecnologías y herramientas que están

desarrolladas para trabajar con XML. XHTML 1.0 se convirtió en la recomendación de World Wide Web Consortium (W3C) en enero del 2000. Aunque XHTML es compatible regresivamente con los navegadores Web actuales y no es muy diferente a HTML 4.01, muchos desarrolladores todavía se resisten a cambiar porque no ven el valor agregado que ofrece una herramienta con un grupo más estricto de especificaciones de sintaxis. [42]

1.5.13 XSL.

XSL está formada por dos lenguajes: XSLT (lenguaje de hojas extensibles de transformación), que permiten generar salidas desde bases de datos XML, incluir etiquetas XHTML y estilos CSS y las XSLFO, que son las hojas extensibles de formato, aún hoy, no soportadas por navegador alguno, se pueden usar las CSS, 100% compatibles, en su reemplazo. [43]

1.5.14 WSDL (Web Services Description Languages)

El lenguaje de descripción de servicios Web WSDL permite aprovechar las ventajas derivadas de SOAP al proporcionar un medio de colaboración sencillo entre los proveedores de servicios Web y los usuarios de los mismos. Con WSDL, se puede automatizar la generación de servidores proxy para los servicios Web de una forma totalmente independiente del lenguaje y la plataforma. Al igual que el archivo IDL de COM y CORBA, el archivo WSDL es un contrato entre el cliente y el servidor. WSDL ha sido diseñado para que pueda expresar enlaces a protocolos distintos de SOAP. WSDL permite la especificación de documentos para la transmisión con SOAP. [44]

WSDL es un formato XML que describe los servicios de red como un conjunto de puntos finales que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un punto final de red. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es extensible, lo que permite la descripción de puntos finales de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse. [45]

1.5.15 SOAP (Simple Object Access Protocol).

Es un protocolo basado en XML que le permite a las aplicaciones intercambiar información sobre HTTP. Es un protocolo estándar creado por el 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. [46]

1.5.16 RUP (Proceso Unificado de Desarrollo)

El Proceso de Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un software. Sin embargo, RUP es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto. El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema o software en construcción está formado por componentes interconectados a través de interfaces. [47]

RUP utiliza el UML (Unified Modeling Language, Lenguaje Unificado de Modelado) para representar todos los esquemas del sistema de software a desarrollar, de hecho UML es una parte esencial de RUP pero sus verdaderos aspectos definitorios se resumen en tres frases claves: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. Es decir, un sistema de software se define para dar servicios a sus usuarios, por lo tanto para construirlo con éxito se debe conocer que es lo que los futuros usuarios necesitan así como que para llevar a cabo el proceso de desarrollo es conveniente dividir el trabajo en partes pequeñas o miniproyectos, representando cada uno de ellos lo que se conoce como iteración. [48]

1.5.17 UML (Unified Modeling Language)

UML (Unified Modeling Language) o Lenguaje de Modelación Unificado es un lenguaje gráfico para especificar, construir, visualizar y documentar las partes o artefactos (información que se utiliza o produce mediante un proceso de software). Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque Orientado a Objetos, utilizándose también en el diseño Web. UML usa procesos de otras metodologías, aprovechando la experiencia de sus creadores, eliminó los componentes que resultaban de poca utilidad práctica y añadió nuevos elementos. [49]

UML es un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, que no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. [50]

De forma general las principales características son:

- ✓ Lenguaje unificado para la modelación de sistemas
- ✓ Tecnología orientada a objetos
- ✓ El cliente participa en todas las etapas del proyecto
- ✓ Corrección de errores viables en todas las etapas
- ✓ Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor. [51]

UML desde finales de 1997 es un lenguaje de modelado orientado a objetos estándar, de acuerdo con el Object Management Group, siendo utilizado diariamente por grandes organizaciones como: Microsoft, Oracle, Rational. [52]

1.5.18 Rational Rose Enterprise Edition.

Rose es una herramienta Case con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. [53]

.Rational Rose permite completar una gran parte de las disciplinas (flujos fundamentales) del proceso unificado de Rational (RUP), en concreto:

- ✓ Modelado del negocio
- ✓ Captura de requisitos (parcial)
- ✓ Análisis y diseño (completo)
- ✓ Implementación (como ayuda)
- ✓ Control de cambios y gestión de configuración (parte). [54]

1.6 Conclusiones.

En este capítulo se realizó una profundización en los conceptos y definiciones necesarias para comprender el proceso de desarrollo del modelado de RCIE; este se acoge a la arquitectura definida por el MINSAP, para dar cumplimiento al programa general de informatización del SNS. Se realizó además un análisis de las tecnologías, lenguajes y herramientas que se tendrán en cuenta para dar cumplimiento al proceso de ingeniería del módulo RCIE.

CAPÍTULO 2

CARACTERÍSTICAS DEL SISTEMA

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

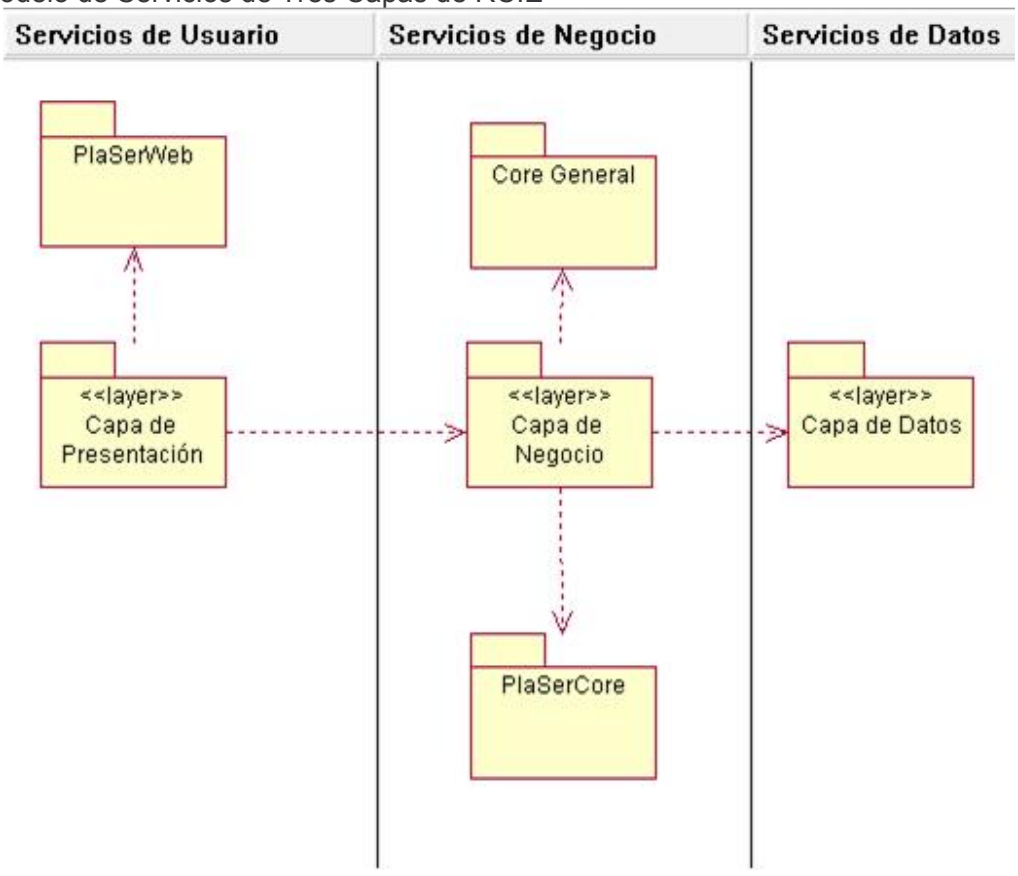
En este capitulo se analiza la arquitectura SOA desarrollada hasta el momento en el modulo RCIE perteneciente al proyecto APS. En el mismo se hará una panorámica de la propuesta de solución que brinda el Plug-in SOA-RUP para cualquier proyecto que use esta arquitectura.

2.1 Diseño actual de la arquitectura en el módulo RCIE.

2.1.1 Arquitectura de tres capas.

El Registro CIE está dividido en 3 Capas (Capa de Presentación, Capa de Negocio y Capa de Datos) y reutiliza el componente PlaSer. Para ello se diseñó un modelo de servicios con la estructura representada en el siguiente diagrama: [55]

Figura 2.1 Modelo de Servicios de Tres Capas de RCIE



Este modelo de servicios propone:

- ✓ **Capa Servicio de Usuario:**

Está compuesto por la capa de presentación y el componente PlaSerCliente. Este se encuentra relacionado con los servicios de interfaces que se le muestra al usuario y por tanto la transformación de XML.

- ✓ **Capa Servicio de Negocio:**

Compuesto por Corel General, la capa de Negocio y el componente PlaSerCore.

- **Core General:** Contiene clases utilizables por todos los módulos de APS, que permiten la impresión de la información con extensión .pdf y .xls.
- **Capa de Negocio:** Contiene las clases del negocio.
- **PlaSerCore:** Está relacionado con la conexión a la base de datos y al acceso de los métodos del negocio.

- ✓ **Capa Servicio de Datos:**

Compuesto por la capa de datos que contiene las clases persistentes del módulo.

Se han hecho todas las realizaciones de cada uno de los casos de uso en los flujos de negocio y análisis y diseño que propone RUP; esto incluye: diagramas de actividades, diagrama de clases del análisis, diagramas de secuencias del análisis, diagrama de clases del diseño y diagramas de secuencias del diseño.

De estos casos de uso el más crítico a tener en cuenta como parte de la arquitectura SOA propuesta es el de **Búsqueda en CIE**, el mismo es representado por las clases controladoras “ficticias” **CC_Busqueda** y **CC_BusquedaAvanzada_RealizarBusqueda** de la capa de negocio, ellas encapsulan todos los métodos relacionados con este funcionamiento, los cuales a su vez constituyen servicios que se brinda a los módulos: Registro de Actividades Diarias, Registro de Partos y Nacimientos, Registros de Fallecidos, Registro de Autopsias, Registro de Enfermedades de Declaración Obligatoria, Registro de Problemas de Salud de la Atención Primaria y Registro de Población.

2.2 RCIE como proveedor de Servicios

La automatización de CIE como se ha explicado anteriormente constituye grandes beneficios en los demás módulos del SIAP, pues el mismo brinda servicios que facilitarán el trabajo con ellos. A continuación se muestra en la siguiente tabla una relación de los servicios brindados por este componente:

Tabla 2.1 Relación de Servicios que brinda RCIE a otros módulos

Módulos de APS	Relaciones
Registro de Enfermedades de Diagnosticación Obligatoria (**REDO**)	Buscar diagnósticos configurados como EDO y seleccionar uno (que devuelva el código y el nombre). Buscar diagnósticos para seleccionar uno (que devuelva el código y el nombre). Esto se usa en el Codificador EDO.
Registro de Actividades Diarias (**RAD**)	Buscar diagnósticos para seleccionar uno (que devuelva el código y el nombre). Tener en cuenta si se van a configurar los grupos para atención primaria (policlínico, consultorio y hospitales rurales), que la búsqueda será en esos grupos. Buscar diagnósticos para seleccionar uno (que devuelva el código y el nombre). Tener en cuenta si se van a configurar los grupos para atención primaria (policlínico, consultorio y hospitales rurales), que la búsqueda será en esos grupos.
Registro de Partos y Nacimientos (**RPN**)	Buscar diagnósticos para la Complicación del parto y seleccionar uno (que devuelva el código y el nombre).
Registro de Fallecido (**RFALL**)	Registrar Causas de Muerte.
Registro de Autopsia (**RAUT**)	Realizar Diagnósticos.

Registro de Población (**RPOB**)	Registrar las causas de ingreso
Registro de Problemas de Salud de la Atención Primaria (**RPSAP**)	Adaptar los problemas de salud internacionales del CIE a la atención primaria de cuba

De todos los módulos antes mencionados solamente se ha concluido su elaboración o se encuentra ya en fase terminal, los siguientes:

- ✓ RPSAP (ver Anexo 7)
- ✓ REDO (ver Anexo 8)
- ✓ RPOB

Los servicios que provee RCIE a estos se comporta de la siguiente forma:

Los tres tienen dentro de la capa de negocio métodos denominados **agentes** dentro de los cuales se hace la petición para acceder a los métodos del negocio de RCIE. Esta petición se realiza mediante un objeto de PlaSerClient y se instancia a una función (call) dentro de la cual se le introducen como parámetros el método del negocio al cual se quiere acceder y los criterios de búsqueda. Para esta petición lo que se envían son mensajes SOAP. Luego RCIE procesa estos datos, se conecta a la base de datos mediante la clase dbz y devuelve la respuesta a la función (call), los valores de respuesta son entregados mediante un SOAP-value.

De forma general la relación entre los métodos del negocio de RCIE y los agentes de cada módulo consumidor quedan mostrados en las siguientes tablas.

Tabla 2.2 Relación ficheros agentes del consumidor RPSAP con los métodos del proveedor RCIE

Proveedor	**Consumidor**
RCIE	**RPSAP (agentes)**
BuscarEnfermedad BuscarSubCategoria	BuscarEnfermedadCIE.php
BuscEnfXCap BuscSubCatXCap	BuscarEnfermedadXCapitulo.php
BuscEnfXGrup BuscSubCatXGrup	BuscarEnfermedadXGrupo.php

BuscEnfXCat BuscarSubCategoria	BuscarEnfermedadXCategoria.php
BuscarGrupo	BuscarGrupoCIE.php
BuscarCategoria	BuscarCategoriaCIE.php
BuscCatXCap	BuscarCategoriaXCapitulo.php
BuscarCapitulo	ListarCapitulosCIE.php
BuscarSubCategoria	BuscarSubCategoriaCIE.php
BuscSubCatXCap	BuscarSubCategoriaXCapitulo.php
BuscSubCatXGrup	BuscarSubCategoriaXGrupo.php

Tabla 2.3 Relación ficheros agentes del consumidor REDO con los métodos del proveedor RCIE

Proveedor	**Consumidor**
RCIE	**REDO (agentes)**
BuscarEnfermedad	BuscarEnfermedadCIE.php
BuscEnfXCap	BuscarEnfermedadCIEXCapitulo.php
BuscEnfXGrup	BuscarEnfermedadCIEXGrupo.php
BuscEnfXCat	BuscarEnfermedadCIEXCategoria.php
BuscarGrupo	BuscarGrupoCIE.php
BuscarCategoria	BuscarCategoriaCIE.php
BuscarCapitulo	BuscarCapituloCIE.php
BuscarSubCategoria	BuscarSubCategoriaCIE.php

Tabla 2.4 Relación ficheros agentes del consumidor RPOB con los métodos del proveedor RCIE

Proveedor	**Consumidor**
RCIE	**RPOB (agentes)**
BuscarEnfermedad BuscarSubCategoria	BuscarProblemasSaludCIE.php
BuscEnfXCap BuscSubCatXCap	BuscarProblemasSaludXCapitulo.php
BuscEnfXGrup BuscSubCatXGrup	BuscarProblemasSaludXGrupo.php

BuscEnfXCat BuscarSubCategoria	BuscarProblemasSaludXCategoria.php
BuscarGrupo	BuscarGrupoCIE.php
BuscarCategoria	BuscarCategoriaCIE.php
BuscarCapitulo	BuscarCategoriaXCapitulo.php
BuscarSubCategoria	BuscarSubCategoriaCIE.php
BuscSubCatXCap	BuscarSubCategoriaXCapitulo.php
BuscSubCatXGrup	BuscarSubCategoriaXGrupo.php

2.3 RCIE como consumidor de Servicios (con SAAA)

Cualquier módulo perteneciente al RIS, como lo es el presente caso de estudio, tiene un sistema de seguridad y control para la gestión de usuarios que accedan a cualquier componente del mismo, denominado SAAA (Single Authentication Authorization and Account).

Cuando un usuario desea trabajar sobre un componente, este introduce su nombre y contraseña, parámetros de entrada al fichero login.php implementado para la página principal del SISalud (Sistema Informatizado de la Salud), el cual le envía estos datos mediante un xml a autentificar.php del negocio de SAAA, y según quien se haya logueado, este método le hace llegar un xml al módulo en cuestión, que en este caso es RCIE, los derechos de acceso, de gestión y tipo de usuario. (Ver Anexo 9)

Existen tres tipos de usuarios: administrador, que puede editar cualquier tipo de información a cualquier nivel, el editor con capacidad de poder gestionar la información del módulo al que pertenece, y el visualizador solo con la posibilidad de consultar la información del módulo al que pertenece.

Todo lo relacionado con lo descrito anteriormente sobre el papel que juega RCIE tanto como abastecedor o consumidor en la transferencia de servicios, no se encuentra modelado o correctamente modelado en la ingeniería de software realizada para representar a este módulo y los demás relacionados con él. Pues no se realizó ni un modelo de servicios, ni un adecuado modelo de diseño que incluya como artefacto componentes de servicios, proposición que hace RUP para SOA, según el Plug-in.

2.4 Propuesta de solución del Plug-in SOA-RUP para aplicaciones que utilizan SOA.

2.4.1 Vista Global

2.4.1.1 Actualización de RUP para SOA

El Plug-in SOA constituye una actualización para el Rational Unified Process (RUP) con el propósito de introducir a una guía para el Arquitecto y Diseñador de un software en el desarrollo de un Modelo de Servicio, un modelo que representa un portafolio de servicios que pueden ser usados como base para las actividades de implementación de 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 y requisitos funcionales y servicios requeridos para soportar un proceso.

2.4.1.2 Alcance

En este Plug-in se describen nuevas actividades y artefactos que implican cambios al análisis y diseño creando un modelo de servicio con sus propios artefactos y estereotipos. De lo que se trata es de actualizar el proceso unificado del software (RUP) con las nuevas características que implica el uso de SOA.

2.4.1.3 El cambio

Las siguientes áreas son las afectadas por esta actualización. La organización está hecha por lo Detalles de Flujos de Trabajo que propone RUP.

- ✓ Concebir un nuevo proyecto
- ✓ Identificar Procesos de Negocio
- ✓ Explorar la automatización de proceso.
- ✓ Manejar el alcance del sistema
- ✓ Realizar la síntesis arquitectónica
- ✓ Refinar la arquitectura
- ✓ Diseñar componentes
- ✓ Componentes del diseño (realización)
- ✓ Diseño de la Base de Datos
- ✓ Implementar Componentes

2.4.2 Disciplinas

Las fases o disciplinas que propone RUP para SOA son las siguientes:

- ✓ Modelamiento del Negocio
- ✓ Levantamiento de Requisitos
- ✓ **Análisis y Diseño**
- ✓ Implementación
- ✓ Pruebas
- ✓ Despliegue
- ✓ Control de Configuración y Cambio
- ✓ Administración de Proyecto
- ✓ Ambiente.

Teniendo todo el centro de atención la disciplina de análisis y diseño, debido a que es ahí donde más se realizan los cambios necesarios en la metodología común de RUP que tienen que ver con una aplicación de arquitectura orientada a servicio.

2.4.3 Análisis y Diseño. Flujos de Trabajo

Los flujos de trabajo propuestos por RUP para SOA son los siguientes:

- ✓ **Definir la Arquitectura Candidata**

Este flujo de trabajo persigue las siguientes metas:

- Crear una descripción inicial de la arquitectura del sistema:
- Definir los elementos arquitectónicamente significativos que serán usados como base para el análisis.
- Definir un sistema inicial de mecanismos de análisis.
- Definir el sostén y organización iniciales del sistema.
- Definir las realizaciones de casos de usos que son dirigidas en la iteración actual.
- Identificar las clases del análisis de los casos de usos arquitectónicamente significativos.
- Actualizar las realizaciones de casos de usos con las interacciones de las clases del análisis.

- ✓ **Realizar la Síntesis Arquitectónica**

Este flujo de trabajo trata de mostrar lo ya se ha realizado, o lo que parece estarlo, es una solución que satisface los requerimientos arquitectónicamente significativos, mostrando así que el sistema, como fue previsto, es factible.

- ✓ **Refinar la Arquitectura**

Dentro de las actividades desarrolladas por el arquitecto de software que propone SOA-RUP en este flujo ya se comienza a tener en cuenta el diseño de artefactos de Modelo de Servicios y de los Componentes de Servicios que se utilizan en una arquitectura SOA.

Los trabajadores que se propuestos relacionados en este flujo de trabajo son el arquitecto de software y el revisor técnico, recayendo la mayor responsabilidad sobre el primero. El mismo realiza las siguientes actividades, en las cuales tienen como entradas una serie de artefactos y a partir de los de salida es que se refina la arquitectura:

- Describir la Distribución
- Describir el tiempo de corrida de la arquitectura
- Identificar elementos del diseño
- En esta actividad se tiene como artefacto entrante más importante el modelo de servicio a diseñar, apoyado por los artefactos: especificaciones suplementarias, pautas específicas del proyecto, documento de la arquitectura del software, modelo de análisis, modelo de diseño y las clases del análisis; teniendo como resultado un modelo de diseño más refinado junto con nuevos artefactos generados como: protocolo a tener en cuenta, encapsulamiento, eventos, señales, subsistemas de diseño, modelo de diseño, clases del diseño, diseño de paquetes, interfaces y componentes de servicios.
- Identificar Mecanismos de Diseño
- Incorporar Elementos de Diseños Existentes
- Identificar servicios

En estas tres últimas actividades se van refinando y perfeccionando todo lo relacionado con los modelos de servicios y componentes de servicios, así como los demás artefactos mencionados.

- ✓ **Diseñar la Base de Datos**

Este flujo de trabajo persigue las siguientes metas:

- Identificar Clases Persistentes
- Diseñar una estructura apropiada de la base de datos
- Diseñar mecanismos y estrategias para guardar y recuperar datos persistentes
- La Base de datos y el almacenamiento de los datos persistentes son implementados.

En este flujo de trabajo los trabajadores propuestos son el revisor técnico, el diseñador de la base de datos y el diseñador, cayendo la mayor responsabilidad sobre el segundo.

La actividad fundamental en aquí es el Diseño de Clases, en la cual se obtienen como artefactos resultantes un modelo de diseño, un diseño de clases y componentes de servicios perfeccionados.

✓ **Analizar comportamiento.**

Características del flujo de trabajo:

Este flujo de trabajo se realiza cada vez que haya una iteración donde existan comportamientos de requisitos que deben ser analizados y diseñados.

Aquí se debe determinar si las clases del análisis encajan en la lógica de la arquitectura. Las clases del análisis pueden determinarse para pertenecer a subsistemas existentes o bien requerir la creación de nuevos subsistemas.

✓ **Diseñar componentes**

Este flujo de trabajo persigue las siguientes metas:

- Refinar las definiciones de elementos diseñados incluyendo cápsulas y protocolos.
- Refinar y actualizar las realizaciones de casos de usos basados en nuevos elementos de diseño identificados.

En este flujo de trabajo los trabajadores a desarrollarlo son el diseñador, revisor técnico, diseñador de cápsula y el diseñador de prueba, recayendo más responsabilidades sobre el primero.

Las actividades principales son:

- **Diseño de encapsulamiento**

Dentro de la cual se tiene como uno de sus pasos fundamentales definir las transiciones de estado entra máquinas, donde una vez que se definan los estados se debe considerar las transiciones entre ellos.

- **Diseño de clases**

En esta actividad se perfeccionan los componentes de servicio. Donde al diseñar las clases controladoras se debe tener en cuenta la distribución y comportamiento del sistema: la necesidad de correr partes de la aplicación en diferentes nodos o en diferentes espacios de procesos, introduce a la especialización de elementos del modelo de diseño. Esta especialización a menudo está acompañada por la adición de objetos de control y el comportamiento de la distribución de las clases interfaces y entidades hacia las clases de control. Haciendo esto, las clases interfaces migran hacia la posibilidad de proporcionar servicios de interfaz de usuario, las clases entidades se mueven hacia la creación del servicio de datos y las clases controladoras proporcionan el resto.

- **Diseño de subsistema**

En esta actividad se obtiene como artefacto resultante componentes de servicios refinados.

✓ **Diseñar Servicios**

El propósito de este flujo de trabajo es proporcionar especificaciones detalladas del comportamiento de servicio y modelar la carpeta de servicio en lo que refiere a los proveedores de servicio y particiones.

Este flujo de trabajo persigue las siguientes metas:

- Refinar las definiciones de elementos del diseño de servicios, proporcionando especificaciones detalladas del comportamiento de ellos y modelando la carpeta de servicios en términos de proveedores y particiones.
- Refine y actualizar las realizaciones de casos de uso basadas en nuevos elementos del diseño de servicios identificados (es decir guardando las realizaciones del usar-caso actualizadas)
- Revisar el desarrollo del diseño.

Se aplica principalmente en la fase de elaboración y se repite en las fases de construcción y transición.

Para este flujo de trabajo se propone al diseñador y su actividad única y fundamental es el diseño de servicios teniendo como artefactos de entrada el modelo de diseño, pautas del proyecto, servicios, modelos de servicios, especificación de servicios, documento de arquitectura de software y especificaciones adicionales, y tienen como artefactos resultantes los modelos de servicios, diseños de servicios, servicios y especificaciones de servicios más refinados. Esta actividad acoge los siguientes pasos importantes:

- ✓ **Reusar la Carpeta de Servicio de la empresa.**

Una de las ventajas más conocidas al utilizar una arquitectura orientada a servicios es su capacidad para que estos representen activos reutilizables a través 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 lista determinada de estos.

Cuando se comienza un proyecto es importante saber si están desarrollando los servicios como parte de una lista o si se está desarrollando la funcionalidad de aplicaciones que ellos utilizan. En cada caso el uso de la lista tiene diversas implicaciones; el diseñador de servicios describe su especificación de servicio y la publica como parte de la lista, esta especificación permite que los desarrolladores de la aplicación entiendan los requisitos de interacción para el servicio. El implementador utiliza la misma especificación para desarrollar unas o más implementaciones del servicio, asegurando que la implementación es coherente a la especificación.

- ✓ **Utilizar Mecanismos y Patrones de Diseño.**

Es buena práctica organizar las ideas y desarrollar diferentes visiones o puntos de vista de un sistema y comprobar cómo los servicios que se está desarrollando se ajustan o no a los diferentes puntos de vista. Definir una organización lógica para estos puntos de vista es muy importante. La asignación de un servicio en una visión no implica propiedad (en sentido de UML); es decir, que el mismo servicio puede participar en visiones o puntos de vista múltiples.

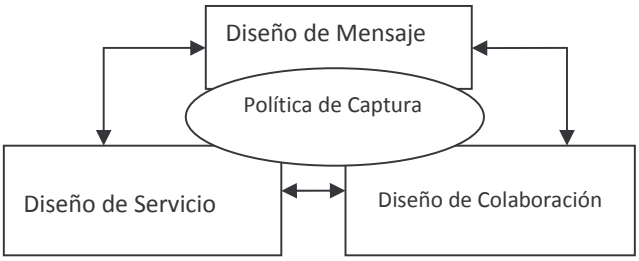
- ✓ **Describir los Elementos de Servicio.**

Para modelar sistemas de software es necesario definir una cantidad X de puntos de entradas a cualquier modelo, además de definir la una metodología aplicable al negocio. En la mayoría de los casos estos puntos de entrada son 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 qué se quiere hacer y asegurar el éxito del proyecto.

Una pequeña cantidad de estas operaciones esta dirigida a desarrollar soluciones orientadas a servicios; el siguiente diagrama 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.

Figura 2.2 Actividades de diseño de Elementos de Servicio.



- ✓ **Diseño del Mensaje.**

En el diseño de mensajes la mayor preocupación esta en el dominio del servicio, técnicas tales como: ingeniería de dominio o análisis y diseño orientado a objetos proporcionan mayor comprensión en el desarrollo de los modelos abstractos de dominio, produciendo generalmente modelos altamente reutilizables para el esquema del mensaje. El diseño de servicio es generalmente una actividad secundaria aunque se hace a veces en paralelo.

- ✓ **Diseño de Servicios.**

El diseñador expone un servicio o sistema de servicios de la funcionalidad esperada del negocio o de la aplicación. En este caso, no se conoce necesariamente lo que elegirá el cliente, todo lo que conocemos es que ellos esperan recibir un servicio de manera rápida y eficiente, por tanto los mensajes tienden a ser secundarios y se convierten en respuesta a los requisitos de una operación determinada.

- ✓ **Diseño de Colaboración.**

El diseño de colaboración está enfocado a la colaboración de dos o más servicios; ésta es una vista de procesos y se relaciona con un negocio más tradicional que modela una actividad determinada del desarrollo del software. Se satisfacen roles en la colaboración y la especificación del servicio y es por tanto un sistema de responsabilidades definidas por el rol a través de unas o más colaboraciones.

2.4.4 Política de identificación y captura

La política de identificación y captura del Plug-in propone la privacidad que se debe tener en cuenta a la vez que un mensaje de servicio es transmitido a un consumidor.

El diagrama siguiente demuestra la asociación de política con los elementos del Modelo de Servicio.

Figura 2.3 Política de identificación del Modelo de Servicios.

2.4.5 Describir dependencia de servicios

Tipos de dependencia en el modelo de servicio:

- ✓ Relación entre un servicio y los Proveedores de Servicios
- ✓ Relación entre un servicio y la Especificación de Servicios
- ✓ Relación entre un servicio y cualquier Especificación de Servicio
- ✓ Relación entre un servicio y cualquier Canal de Servicio que lo conecte a otros servicios (y por consiguiente al otro servicio al final del canal.)
- ✓ Relación entre un servicio y cualquier Partición de Servicio

Mientras que en una parte del modelo de servicio 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 realiza al reutilizar un servicio, particularmente si hay múltiples implementaciones.

Aquí se debe tener en cuenta que la herramienta de software fundamental para el modelado se servicios es el **Rational Software Architect**, que proporciona los siguientes pasos:

- ✓ Crear un nuevo proyecto
- ✓ Crear nuevo modelo de diseño de servicios
- ✓ Crear mensajes para los servicios
- ✓ Crear un nuevo servicio para el modelo de bibliotecas.
- ✓ Crear una nueva partición de servicios.

Todos estos pasos están bien especificados en el Plug-in de SOA-RUP, constituyendo una buena guía para ser aplicados en un caso práctico.

2.5 Roles y actividades

SOA-RUP propone dos roles trabajadores fundamentales en el desarrollo del modelado de una aplicación que use arquitectura orientada a servicios. Estos son el diseñador y el arquitecto del software.

- ✓ **Diseñador**

El rol del diseñador es responsable para diseñar una parte del sistema, dentro de los contrastes de los requisitos, arquitectura, y proceso de desarrollo para el proyecto. Su actividad fundamental es la de diseñar los servicios ya especificadas anteriormente.

- ✓ **Arquitecto de Software**

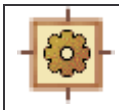
Este rol es el responsable para la arquitectura del software que incluye decisiones técnicas importantes que tienen que ver con el diseño global y la aplicación para el proyecto.

2.6 Artefactos

A continuación se describen en las siguientes tablas todos los artefactos que propone el Plug-in, relacionado con el modelo de servicio.

2.6.1 Modelo de Servicios.

Tabla 2.5 Detalles del Modelo de Servicios

	El Modelo del Servicio es un modelo de elementos que constituye la base de una arquitectura orientada servicios. Es una entrada a las actividades en las disciplinas de implementación y prueba.
Otras relaciones	**Contiene:** • Mensaje • Servicio • Canal de Servicio. • Colaboración de Servicio. • Entrada de Servicio. • Partición de Servicio. • Abastecedor de Servicio. • Especificación de Servicio
Rol	Diseñador
Opcionalidad/Ocurrencia	Obligatorio
Plantillas e Informes que se generan	Reporte: Lista de Servicios. Reporte: Dependencias de Servicio. Reporte: Traceabilidad de meta de servicio.
Representación UML.	Modelo estereotipado como <<Modelo de Servicios>>.

Entrada a las actividades	Salida de Actividades
• Identificar Elementos de Diseño. • Identificar Mecanismos de Diseño. • Incorporar Elementos de Diseño Existentes. • Diseño de Servicio.	• Identificar Elementos de Diseño. • Identificar Mecanismos de Diseño. • Incorporar Elementos de Diseño Existentes. • Diseño de Servicio.

2.6.1.1 Elementos del modelo de servicio

- ✓ **Mensaje**

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

Tabla 2.6 Detalles del mensaje

Mensaje	
Otras Relaciones	Parte de 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.

- ✓ **Servicio.**

Un servicio es un modelo de los elementos de la base de una Arquitectura Orientada Servicio. Un servicio es proporcionado por un abastecedor de servicio y es un caso de una Especificación del Servicio.

Tabla 2.7 Detalles del servicio.

Servicio	
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 <<Servicio de Especificación>>.

- ✓ **Canal de Servicio.**

Un canal de servicio es un elemento del Modelo de Servicio que representa una conexión entre dos servicios, o entre un cliente y un excedente del servidor que deriva de la interacción con el servicio. Es importante entender que el canal no representa ninguna interacción en particular.

Tabla 2.8 Detalles del canal de Servicio

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

- ✓ **Colaboración de Servicio.**

Tabla 2.9 Detalles de Colaboración de Servicios

	La colaboración de servicio es una representación de algunas sesiones de comunicación entre dos o más servicios usualmente encapsulados como un

	nuevo servicio. De esta manera el modelo puede representar servicios cuya implementación es simplemente la colaboración de una sesión de servicios existentes.
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador
Opcionalidad/Ocurrencia:	Opcional
Representación UML	Colaboración, estereotipado como <<Colaboración de Servicio>>. Los participantes en la colaboración pueden solamente ser instancias de Proveedores de Servicios.

- ✓ **Consumidor de Servicios**

Cualquier clasificador (clase, componente, etc.) puede actuar como consumidor de un servicio, y eso incluye otro servicio. Mientras este estereotipo es el más definitivamente optativo que puede ser útil en la identificación de elementos de un modelo—no tiene sus propios servicios--como clientes de servicios. Por otro lado puede ser abordado y no usado.

Tabla 2.10 Detalles de Consumidor de Servicios

Consumidor de Servicios	
Otras Relaciones	Parte de Modelo de Servicios.
Rol	Diseñador
Opcionalidad/Ocurrencia:	Opcional, generalmente no es muy utilizado
Representación UML	Un mensaje no tendrá operaciones o especificaciones de comportamiento definidas.

- ✓ **Entrada de Servicio.**

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

Tabla 2.11 Detalles de Entrada de Servicio

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

- ✓ **Partición de Servicio.**

Una partición de servicio es un elemento modelo que proporciona un agrupamiento lógico para los abastecedores 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.

Tabla 2.12 Detalles de Partición de Servicios

Partición de servicio	
Otras Relaciones	Parte del Modelo de Servicios.
Rol	Diseñador.
Opcionalidad/Ocurrencia:	Opcional
Representación UML	Clase, Componente o Nodo, estereotipado como <<Servicio de

	Partición>>. El Servicio de Partición no tendrá ningunas operaciones o cualidades, no tendrá ningún comportamiento especificado y no realizará ningún interfaz. Cualquier puerto en la partición de servicio será estereotipado como <<Servicio de Entrada>> y cualquier estructura compuesta especificará solamente las piezas que son abastecedores de servicio.

- ✓ **Proveedor de Servicios.**

Un proveedor de servicio agrupa un sistema relacionado de servicios.

Tabla 2.13 Detalles de Proveedor de Servicio

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

- ✓ **Especificación de Servicio.**

Una Especificación de Servicio es un elemento de modelo que provee la especificación estructural y de comportamiento de una instancia de servicio. También puede identificar un grupo de políticas para acceder a él o a su uso.

Actúa como un contrato entre el cliente del servicio y el implementador del mismo, el cliente entiende cómo obrar recíprocamente con un servicio y el implementador entiende el comportamiento esperado de su implementación.

Tabla 2.14 Detalles de Especificación de Servicios

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

2.6.1.2 Reportes.

- ✓ **Paquete de Servicio**

Tabla 2.15 Detalles de Paquete de Servicio

	Este informe contiene una vista de la carpeta del Artefacto: Modelo de servicio.
Artefactos Asociados:	Modelo de Servicio
Más información:	Este informe describe al modelo de servicio comprensivamente, por lo que se refiere a cómo las características técnicas detalladas en él.

- ✓ **Dependencias de servicio.**

Tabla 2.16 Detalles de Dependencias de servicio

	Este informe contiene una vista de la dependencia del Artefacto: Modelo de servicio.
Artefactos Asociados:	Modelo de Servicio

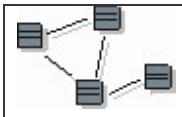
- ✓ **Traceabilidad de los objetivos del servicio**

Tabla 2.17 Detalles de Traceabilidad de los objetos del servicio

	Este informe contiene un servicio a la vista de traceabilidad de metas del Artefacto: Modelo de servicio.
Artefactos Asociados:	Modelo de Servicio

2.6.2 Modelo de Diseño

Tabla 2.18 Detalles del 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 del diseño se utiliza como entrada esencial a las actividades en la disciplina de implementación y prueba.

Otras Relaciones	**Contiene** • Protocolo.  • Cápsula.  • Realización de Caso de Uso.  • Señales.  • Eventos.  • Subsistema de Diseño  • Paquete de Diseño.  • Interfaz.  • Clases de Diseño.  • Clases de Pruebas  • Diseño de Pruebas  • Componente de Servicio. 
Rol	Arquitecto de Software.
Opcionalidad/Ocurrencia	Requerido. Fases de elaboración y de construcción.
Plantillas y Reportes	• Estudio del modelo de diseño • Realización de Casos de Uso • Subsistema de Paquetes de Diseño

	• Reporte de Clase
Representación UML	Modelo estereotipado como <<Modelo de Diseño>>.
Entrada a las actividades • Análisis de la Arquitectura. • Diseño de Clases. • Construcción del concepto de Prueba de la arquitectura. • Diseño de la Base de Datos. • Definir elementos de Prueba • Describir Distribución • Describir el tiempo de corrida de la Arquitectura • Identificar elementos de Diseño • Identificar mecanismos de Diseño. • Identificar ideas de pruebas • Implementar elementos del Diseño • Implementar prueba de Desarrollo • Incorporar elementos de diseño existentes. • Administrar Dependencias • Revisar el diseño	**Salida de Actividades** • Análisis de la Arquitectura. • Diseño de Clases • Describir el tiempo de corrida de la Arquitectura • Identificar elementos de Diseño. • Identificar mecanismos de Diseño. • Incorporar elementos existentes de diseño. • Diseño de Servicio. • Subsistema de Diseño. • Casos de Uso del Diseño.

• Diseño de Servicio. • Estructurar el Modelo de Implementación. • Subsistema de Diseño • Casos de Uso del Análisis • Casos de Uso del Diseño.	

✓ **Artefactos del modelo del diseño**

Tabla 2.19 Detalles del Componente de Servicios

	El componente del servicio es una especialización del subsistema del diseño prevista para el uso en describir la realización de una especificación del servicio. Un componente del servicio puede proporcionar la realización para unos o más servicios por la realización de las especificaciones múltiples del servicio. El sistema de los elementos modelo en el interior del componente representa la realización concreta del contrato estructural, del comportamiento y de la política descrito por estas especificaciones de servicios.
Otras Relaciones	Parte de Modelo de Servicios Subsistema Extendido del Diseño
Rol	Diseñador

Opcionalidad/Ocurrencia	Obligatorio
Representación UML	Componente de UML 2.0. Estereotipado como << Componente de Servicio >>. Notar que UML 2.0 proporciona un estereotipo, dentro del perfil intermedio, llamado <<servicio >>, sin embargo esto simplemente se define como un "Un componente sin estado, funcional (computa un valor) " qué no lleva el significado implicado por este elemento de modelo.

- ✓ **Otros artefactos a tener en cuenta**

Tabla 2.20 Artefactos de modelo del diseño

Icono	Nombre	Representación UML	Descripción
	fachada	Estereotipo en Clase o Componente	Usado para denotar el componente que actúa como una fachada para la implementación del servicio; en general hay un componente fachada para cada Especificación de Servicio realizada.
	mediator	Estereotipo en Clase o Componente	Usado en situaciones dónde puede haber uno o más aplicaciones para un funcionamiento de servicio dado, el mediador se llama por la fachada para identificar y llamar el componente de aplicación correcto.

	Acceso a Datos	Estereotipo en Clase o Componente.	Usado para denotar el acceso al componente de acceso a datos, este componente es responsable del acceso y control de los datos persistentes para la implementación de servicio.

2.7 Herramientas a utilizar que propone SOA-RUP

Esta sección proporciona una guía para el trabajo con RUP con la ayuda de esta nueva herramienta. Propone descripciones de como ejecutar actividades específicas de procesos o pasos, o produce un reporte de un artefacto en particular, usando una o más herramientas.

Las herramientas que se incluyen en esta configuración son Rational Unified Process, **RUP Builder,** Rational Process Workbench, Rational Administrator, Rational Suite AnalystStudio, Rational ClearCase, Rational ClearQuest, Rational ProjectConsole, Rational PurifyPlus, Rational QualityArchitect, Rational RequisitePro, Rational Robot, Rational Rose, Rational Rose RealTime, Rational SoDA, Rational TestManager, Rational Test RealTime, Rational TestFactory, Rational XDE Developer - Java Platform Edition, Rational XDE Developer - .NET Edition; de las cuales en las que más se va a enfocar el estudio de este trabajo son el **Rational Software Architect (RSA)** y Rational Rose debido a que la primera es la fundamental en la construcción del modelo de servicio y la segunda es usada en todos los demás modelos que propone RUP.

Es bueno mencionar que algunas herramientas de Rational aparecen en diferentes versiones de la familia de productos de Rational.

2.8 Conclusiones.

En este capítulo se ha valorado la construcción de la arquitectura en RCIE, profundizando en SOA. Se determina que no se encuentra correctamente diseñada la ingeniería de software desarrollada en este módulo, y con el objetivo de llegar a una posible solución se realizó una panorámica de los pasos, actividades y artefactos que agrega el Plug-in SOA-RUP para aplicaciones de este tipo y que deberán ser aplicadas en el registro para su perfeccionamiento.

CAPÍTULO 3

ANÁLISIS Y DISEÑO DEL SISTEMA

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Todo lo relacionado con lo descrito en el capítulo anterior sobre el papel que juega RCIE tanto como abastecedor o consumidor en la transferencia de servicios, no se encuentra correctamente modelado en la ingeniería de software realizada para representar a este módulo y los relacionados con el.

A continuación se tomarán en cuenta algunos pasos y diagramas a considerar para el desarrollo del modelo de servicio adecuado, teniendo como material de apoyo los estereotipos que propone la herramienta Rational Software Architect para representar los artefactos de servicios:

3.1 Pasos a seguir para realizar un Modelo de Servicios.

- ✓ **Identificación de servicios:**

Es una de las primeras acciones en el modelamiento de una solución orientada a servicio y por consiguiente los errores hechos durante la identificación, pueden influir en el diseño detallado y actividades de implementación. Se tienen una serie de técnicas para llevar a cabo esta actividad; pero fundamentalmente se debe tener en cuenta la conexión con el modelo de análisis del negocio. La conexión más directa entre este y el modelo de servicio permite que los servicios puedan ser vistos para apoyar las necesidades del negocio.

Con las metas del negocio que manejan el mismo es más fácil identificar una alineación entre Servicios y Metas; así será posible listar, para cualquier Especificación de Servicio todas las metas del Negocio que contribuyen a ello, y viceversa para cualquier Meta del Negocio se pueden listar los Servicios actualmente desplegados en la organización. De un Modelo de Análisis del Negocio que incluye Entidades del Negocio y Reglas del Negocio es posible definir servicios que encapsulan reglas del negocio.

- ✓ **Creación del Modelo de Servicio:**

La actividad de Identificación de Servicio introduce un número de técnicas para identificar servicios y operaciones sobre servicios, requeridas para soportar una solución. En el caso de este ejemplo se busca un formulario para la búsqueda en RCIE, la transformación de funcionalidades existentes para un modelo de servicios y específicamente tecnología de implementación del modelo de servicios. El hecho de este primer aspecto es desarrollar un conjunto de Especificaciones de Servicios que proporcionarán el contrato para la implementación de servicios de las funciones antes descritas.

- ✓ **Identificar Particiones de Servicios:**

Una partición de Servicio puede ser una Clase, Componente o Nodo. La partición de servicio no tendrá operaciones o cualidades, no tendrá ningún comportamiento especificado y no realizará ninguna interfaz.

- ✓ **Desarrollo de Especificación de Servicios:**

Se diseñan y crean las especificaciones de servicios que se puedan dar lugar en el sistema para después modelar su futura implementación.

- ✓ **Diseño de Servicios:**

En esta actividad se identifican los proveedores de servicios y las relaciones que existen entre ellos; así como las especificaciones de servicios y servicios que contienen.

- ✓ **Diseño de mensajes:**

En la actividad de identificación de servicios, no se toma en cuenta los mensajes reales intercambiados entre las operaciones descritas en las especificaciones de servicio, pero esto puede ser un acercamiento común para capturar las responsabilidades que difieren del diseño detallado de los mensajes, y en muchos casos este acercamiento se invierte, donde las estructuras del mensaje son conocidas antes de tiempo y agregadas a un grupo de operaciones.

En general existe una gran influencia del modelo del dominio desarrollado como una parte del modelo de análisis, por tanto el modelo de mensajes no se construye de la nada sino que son identificados como un subconjunto del modelo de dominio que puede ser reusado. Se asume también que son datos que transfieren los objetos, estructuras que pasan entre los servicios. Así en lugar de cambiar el modelo de dominio, en caso de necesidad, se crean los mensajes fuera de la estructura de las clases.

- ✓ **Realización de Servicios:**

En esta actividad se realiza finalmente un diagrama por cada especificación de servicio detalladas en la actividad de Desarrollo de Servicio, constituyendo esto las realizaciones de las especificaciones de servicio. Para ello se deben tener en cuenta los siguientes aspectos y elementos:

- **Componente de Servicio:**

Es una especialización del subsistema del diseño prevista para el uso en describir la realización de una especificación del servicio y no necesariamente es un componente de implementación. Un componente del servicio puede proporcionar la realización para unos o más servicios por la realización de las especificaciones múltiples del servicio.

- **Componente Fachada:**

Usado para denotar el componente que actúa como una fachada para la implementación del servicio; en general hay un componente fachada para cada Especificación de Servicio realizada. La fachada comprende la misma interfaz como el propio componente de servicio y mantiene las capacidades básicas para la validación del mensaje antes de pasar la demanda a los componentes para la ejecución. En este caso se estereotipa el componente como << la fachada >> para clarificar.

- **Componente de acceso a datos:**

Usado para denotar el acceso al componente de acceso a datos, este componente es responsable del acceso y control de los datos persistentes para la implementación de servicio.

- El modelo de servicio es un artefacto del diseño-tiempo y como tal no trata directamente con la implementación de servicios. Sin embargo, la actual implementación de un servicio o grupo de servicios se realiza estrictamente por la realización de una especificación de servicio por un componente de servicio. La especificación de servicio proporciona el contrato de implementación, la tecnología o técnicas usadas para implementar el servicio son irrelevantes hasta que el contrato se cumpla.

- Un aspecto importante de la relación entre el Modelo de Servicio y el componente de Modelo de Diseño es que el grupo de Especificaciones de Servicio representa contratos que deben cumplirse, que las operaciones identificadas en las especificaciones debe llevarse a cabo como-es, que los consumidores de servicios están usando a este mismo modelo para entender la interfaz y conducta de los servicios que ellos esperan usar. Como tal hay directa y relación general uno a uno entre la Especificación de Servicio y algunos artefactos de implementación que actúan como el punto de entrada de aplicación inicial para el servicio.

Finalmente se puede concluir que será muy importante para realizar el modelo de servicio relacionado con RCIE, identificar cuáles serían los componentes fachada, de servicios y de acceso a datos en cada realización de especificación de servicios que se desarrolle, sin divorciarse de lo que ya se encuentra

implementado y teniendo en cuenta que por cada especificación de servicios existe un componente fachada y de servicio.

3.2. Modelo de Servicios de RCIE para RPSAP

Considerando las actividades a seguir para el desarrollo de un modelo de servicios y por otra parte que los tres módulos ya implementados que consumen de RCIE, tienen la misma arquitectura y el mismo comportamiento al solicitar un servicio mediante agentes, y que además el consumo es común (el de buscar enfermedades en CIE), se desarrollará por los pasos antes descritos una propuesta de modelo de servicio de RCIE para RPSAP, pudiéndose este aplicar en los restantes componentes con sus agentes específicos.

Identificación de Servicios

Teniendo en cuenta los pasos a seguir para la realización de esta actividad se identificaron los siguientes servicios, tomándolos de la realización del Caso de Uso Buscar en RCIE en el módulo RPSAP. Para ello se consideró que la aplicación de RPSAP busca enfermedades de dos formas en RCIE: Búsqueda General y Búsqueda Específica, dentro de las cuales se agrupan determinados servicios a consumir. Finalmente resultó de la siguiente manera:

- ✓ **Búsqueda General:**

Buscar Enfermedad.
Buscar SubCategoría.
Buscar SubCategoría.
Buscar Grupo.
Buscar Capitulo.

- ✓ **Búsqueda Específica:**

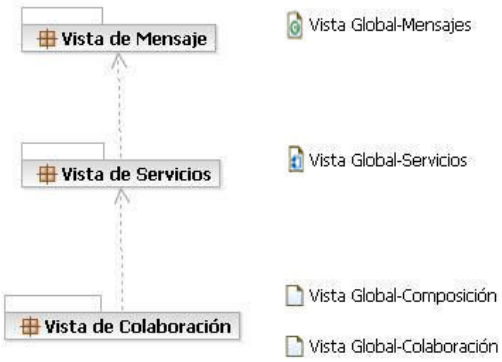
Buscar Enfermedad por Capitulo.
Buscar SubCategoria por Capitulo.
Buscar Enfermedad por Grupo.
Buscar SubCategoria por Grupo.
Buscar Enfermedades por Categorias.
Buscar Categorias por Capítulos.

Creación del Modelo de Servicio

La creación del Modelo de Servicio fue hecha acorde a UML para servicios de software y se utilizaron los estereotipos de clases del Rational Software Architect (RSA), herramienta que incluye un modelo de plantilla como se describe en el siguiente diagrama. Este modelo es identificado como un refinamiento del modelo de análisis. A continuación se muestra un diagrama global que describe las dependencias entre vistas recomendadas por la plantilla del RSA y que es aconsejable utilizar para el ejemplo práctico en cuestión.

Nota: Es bueno esclarecer que para el desarrollo posterior del presente trabajo se produjeron artefactos guiándose por la plantilla propuesta, pero no se modeló en la herramienta, debido a problemas de instalación pues el ejecutable que se tuvo disponible no incluía el Modelado de Diseño de Servicios que propone el Plug-in en su guía para trabajar en el RSA.

Figura 3.1 Modelo de Servicio propuesto por el RSA



Identificar Particiones de Servicios

Para particionar los servicios del sistema se pueden encontrar un número de vías bastante crecido, pero solo se deben tener en cuenta aquellas que sean más críticas en el diseño de la arquitectura. En este caso se llegaron a dos particiones de Servicios que consume la aplicación RPSAP, estas son: Búsqueda General en RCIE y Búsqueda Específica en RCIE, que encapsularán los métodos que piden servicios de RCIE; lo que queda demostrado en el siguiente diagrama:

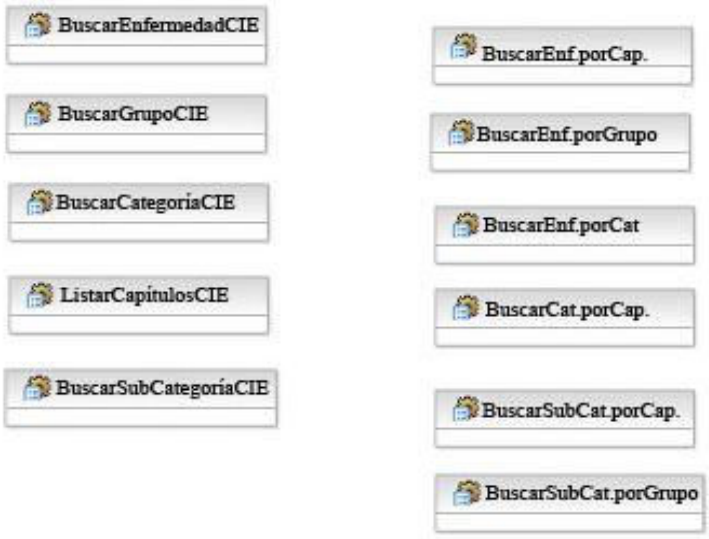
Figura 3.2 Particiones de Servicios

Desarrollo de Especificación de Servicios

Considerando los servicios identificados anteriormente y teniendo en cuenta que el consumo de estos en RPSAP con RCIE es a partir de agentes, se diseñaron especificaciones de servicios determinadas para cada método agente definido, acogiéndolos de esta forma como consumidores.

El siguiente diagrama muestra un árbol vacío de las especificaciones de servicios creadas en esta fase:

Figura 3.3 Especificaciones de Servicios Identificadas

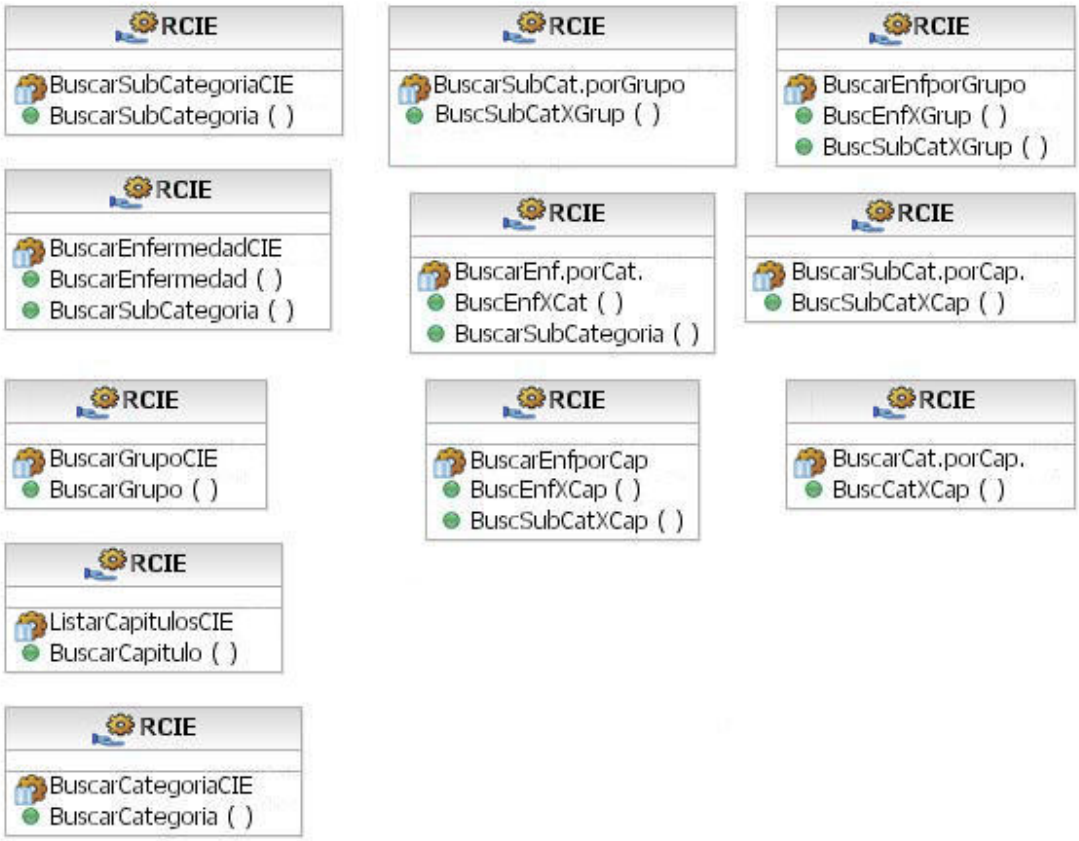


Diseño de Servicios

A continuación se representan los proveedores que en este caso es RCIE, relacionado con las especificaciones de servicios antes descritas y sus funciones. En este punto se debe asumir que es el mismo proveedor, pero para poder establecer posteriormente una relación directa con los agentes

consumidores, se estableció representar al abastecedor de servicios por cada especificación de servicio, resultando de la siguiente forma:

Figura 3.4 Diseño de Servicios



Diseño de mensajes

Para el diseño de mensajes a realizar se tuvo en cuenta que ya los mensajes están implementados, pero los mismos tienen cierta relación con el modelo de dominio descrito para RCIE y a su vez con las operaciones de las especificaciones de servicios antes descritas. En general para cada especificación de servicios en RCIE existen dos ¨ tipos ¨ de mensajes: Request (que contiene los datos introducidos en la petición del servicio) y Response (que contiene los datos que serán entregados).

Por otra parte, es bueno esclarecer que para la representación del diseño de mensajes, los mismos se obtuvieron a partir del wsdl de este módulo, detallados aquí de una forma lineal, por lo que queda configurado el modelo de diseño de la siguiente forma por cada servicio identificado:

Figura 3.5 Diseño de Mensajes del Servicio Buscar Enfermedad.

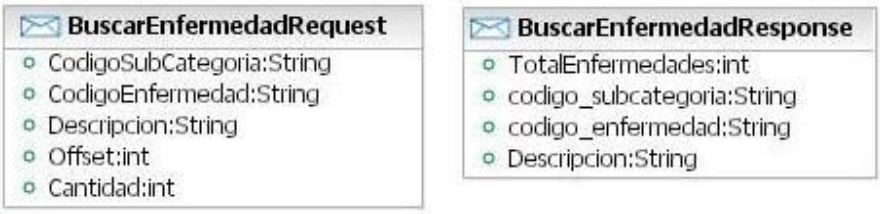


Figura 3.6 Diseño de Mensajes del Servicio Buscar Grupo.

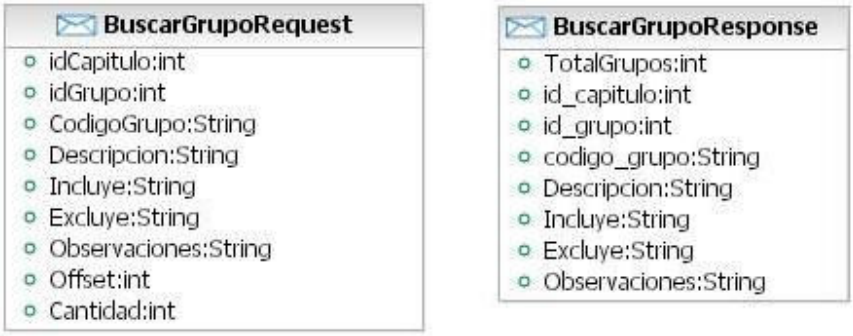


Figura 3.7 Diseño de Mensajes del Servicio Buscar Categoría.

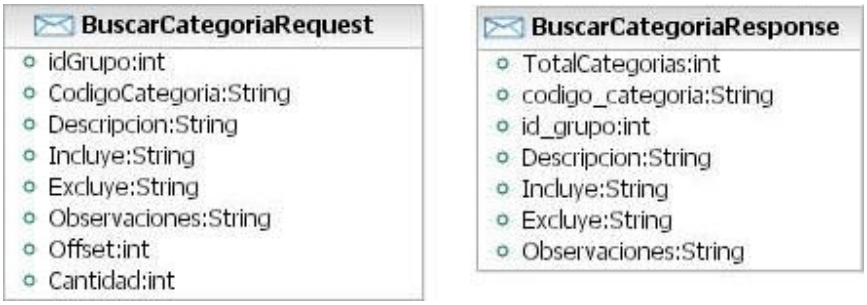


Figura 3.8 Diseño de Mensajes del Servicio Listar Capítulo.

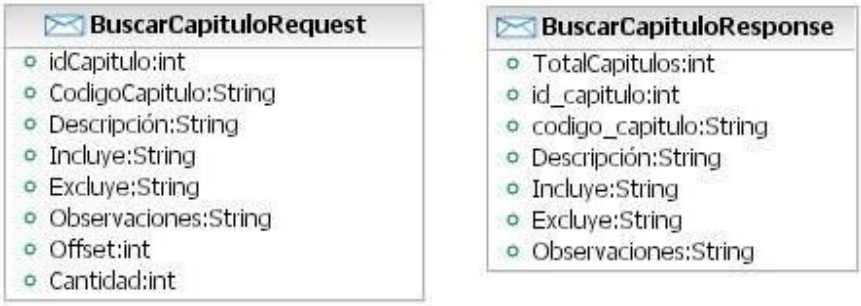


Figura 3.9 Diseño de Mensajes del Servicio Buscar SubCategoría

Figura 3.10 Diseño de Mensajes del Servicio Buscar Enfermedad por Capítulo.

Figura 3.11 Diseño de Mensajes del Servicio Buscar Enfermedad por Grupo

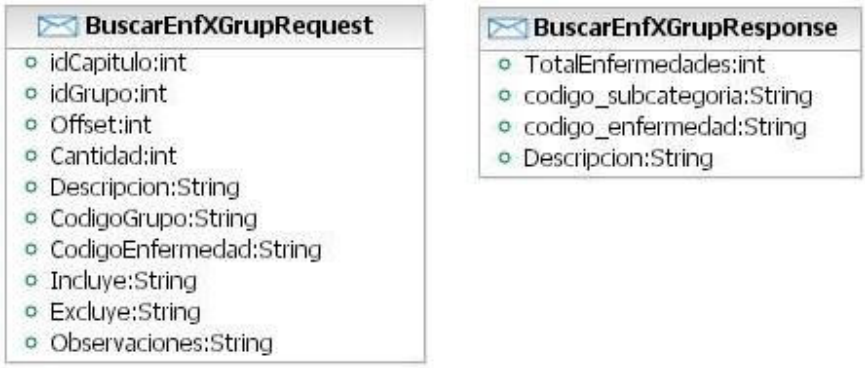


Figura 3.12 Diseño de Mensajes del Servicio Buscar Enfermedad por Categoría

Figura 3.13 Diseño de Mensajes del Servicio Buscar Categoría por Capítulo

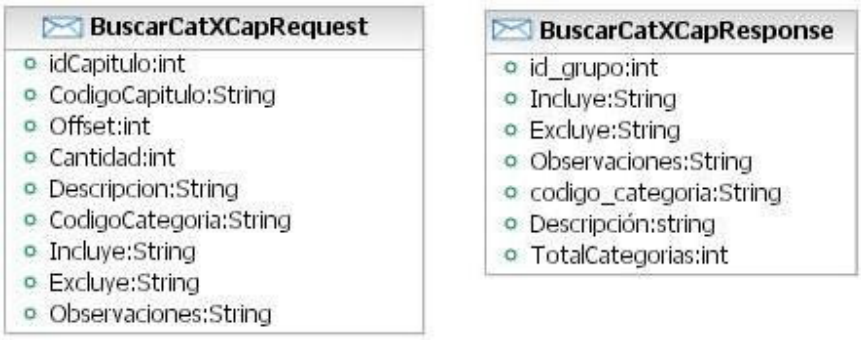


Figura 3.14 Diseño de Mensajes del Servicio Buscar SubCategoría por Capítulo

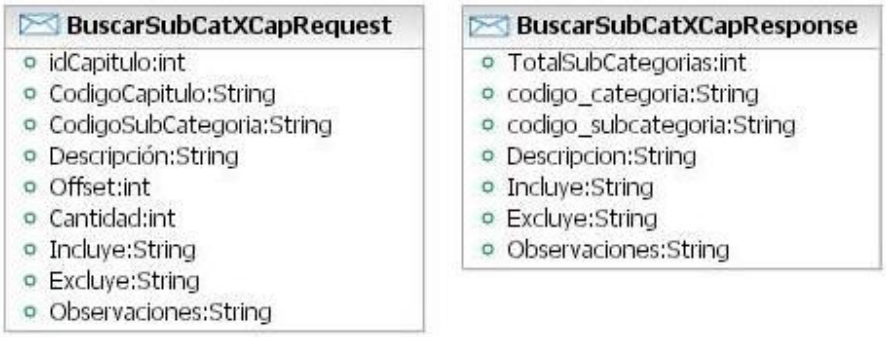
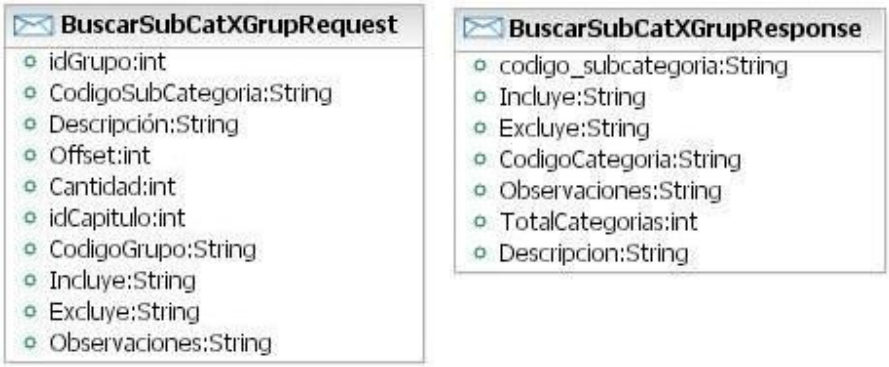


Figura 3.15 Diseño de Mensajes del Servicio Buscar SubCategoría por Grupo



Realización de Servicios

Durante esta actividad se realiza **un diagrama por cada especificación de servicio**, detallada anteriormente. Para la realización de cada especificación se siguieron los siguientes pasos:

- ✓ Por cada especificación de servicio se debe tener un componente de servicios y un componente fachada relacionados a ella.
- ✓ También se deben representar los componentes de conexión a la base de datos.
- ✓ Las relaciones que deben existir entre cada elemento son de uso.
- ✓ Se deben representar los métodos del negocio del proveedor (RCIE) que responden a la implementación del servicio a tratar.

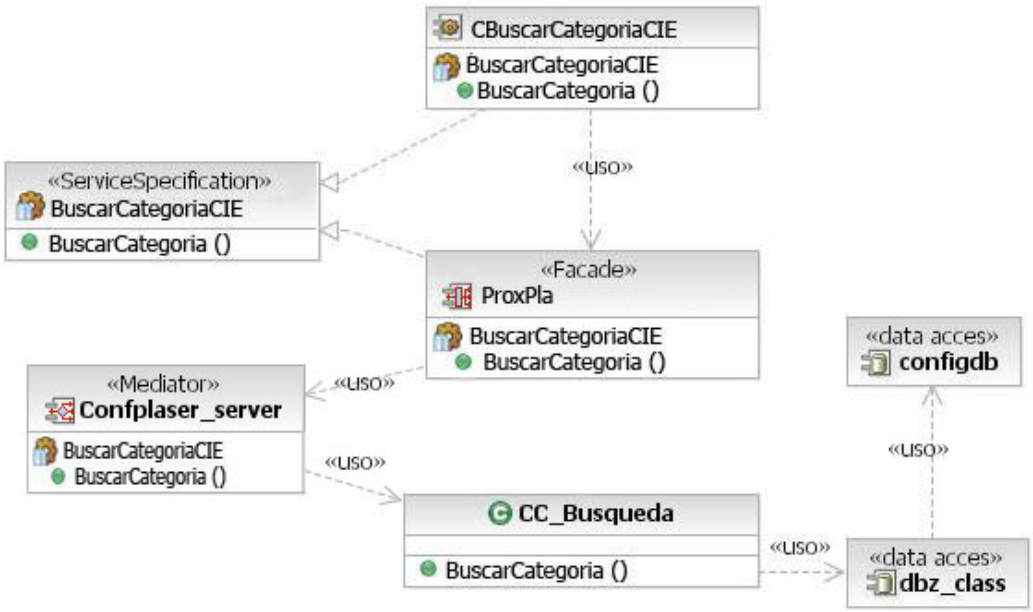
De forma general el desarrollo de cada especificación de servicios establecida, de RCIE para RPSAP, tienen el siguiente comportamiento común:

- ✓ Por cada especificación se generaliza un componente de servicios y un componente fachada.

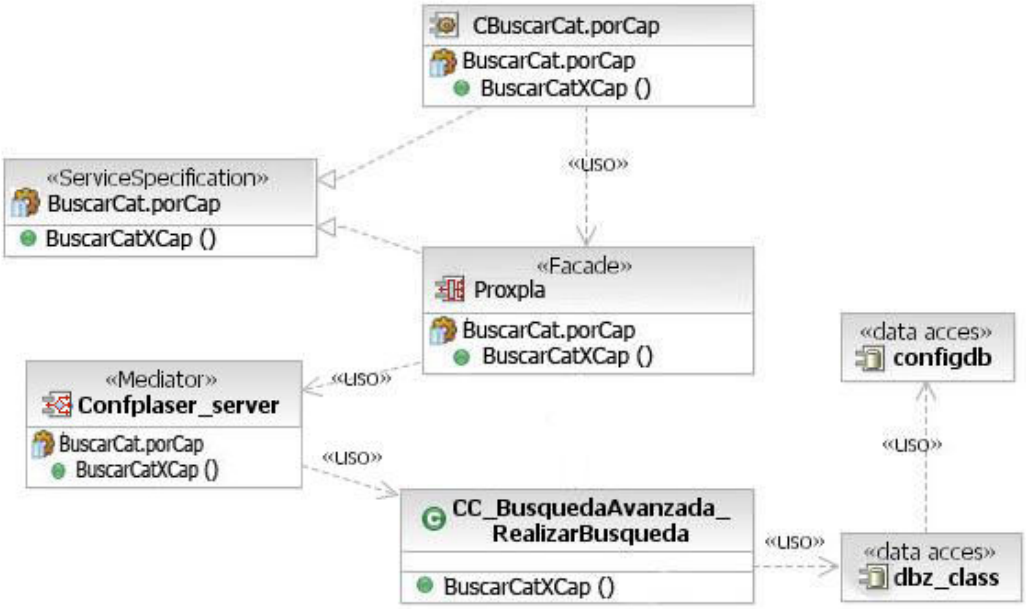
- ✓ Teniendo en cuenta las funcionalidades del componente fachada que plantea el Plug-in se llegó a la conclusión que el componente Proxpla (el cual pertenece al módulo SAAA) cumple con estos requisitos; este componente se encarga de recibir todas las peticiones que hace el sistema y de dar respuestas a las mismas. Configura los métodos empaquetados en un negocio con su tipo, nombre y especificaciones de acceso. Cuando RPSAP hace una petición a RCIE esta petición primero llega al Proxpla, el cual verifica el método y el módulo abastecedor, comportándose así como la primera capa de seguridad para luego dirigirse a la segunda que sería la clase Confplaser_server de RCIE, admitiéndose esta última como el mediador según los requisitos que encierra: el mediador se llama por la fachada para identificar y llamar el componente de aplicación correcto.
- ✓ Dado que el componente ConfPlaser_server contiene al método del negocio de RCIE que se quiere consumir, esta clase lo llama, y a su vez el método en su implementación utiliza a la clase dbz_class para la conexión a la base de datos, considerada esta clase como un componente de acceso a dato.
- ✓ La clase dbz_class implementa el acceso a cualquier tipo de Base de Datos. Dbz utiliza a ConfigDB para obtener la localización de la base de datos y los datos del usuario con derechos de inserción, asumiéndolo así como otro componente de acceso a datos.
- ✓ En cada caso se representó la relación entre la clase mediador con la clase “ficticia” de la capa de negocio diseñada en el modelo de análisis y diseño, pero se debe tomar en cuenta lo que existen son métodos dentro de ficheros los que son llamados por ConfPlaser_server.

Asumiendo lo antes expuesto a continuación se representan en los siguientes diagramas la realización de cada especificación de servicios identificadas para representar el consumo de RPSAP a RCIE:

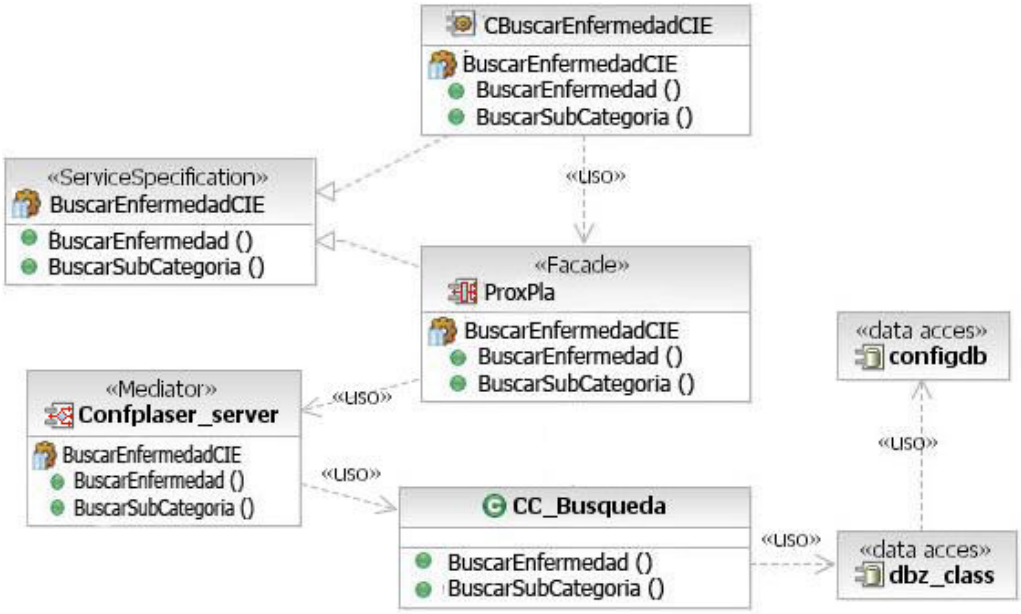
Especificación de Servicio: Buscar Categoría CIE
Figura 3.16 Realización de la especificación de servicio Buscar CategoríaCIE



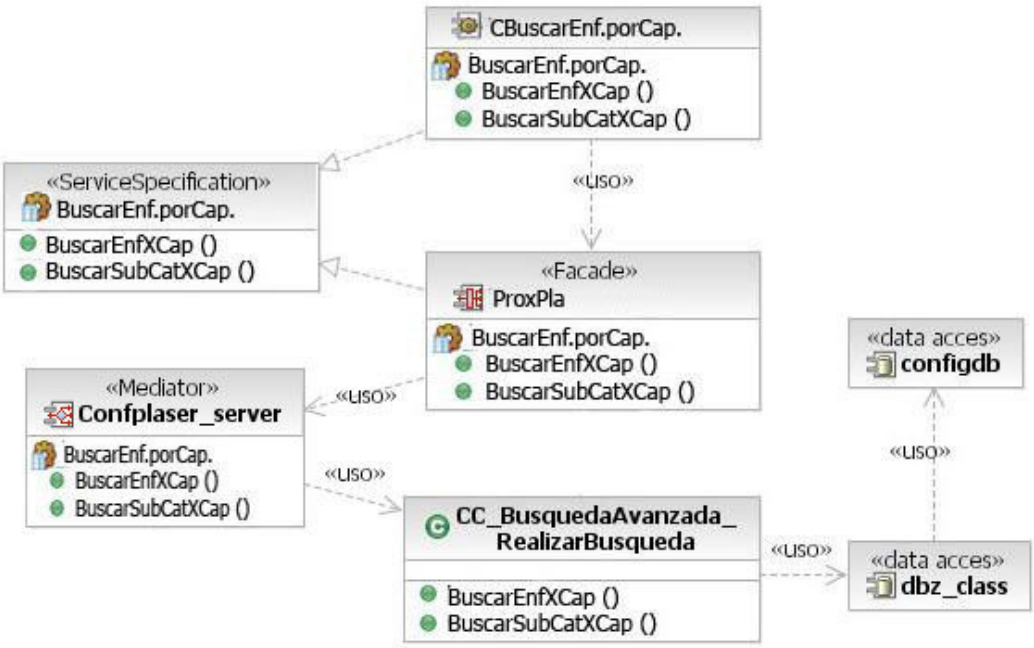
Especificación de Servicio: Buscar Categoría por Capítulo
Figura 3.17 Realización de la especificación de servicio Buscar Categoría por Capítulo



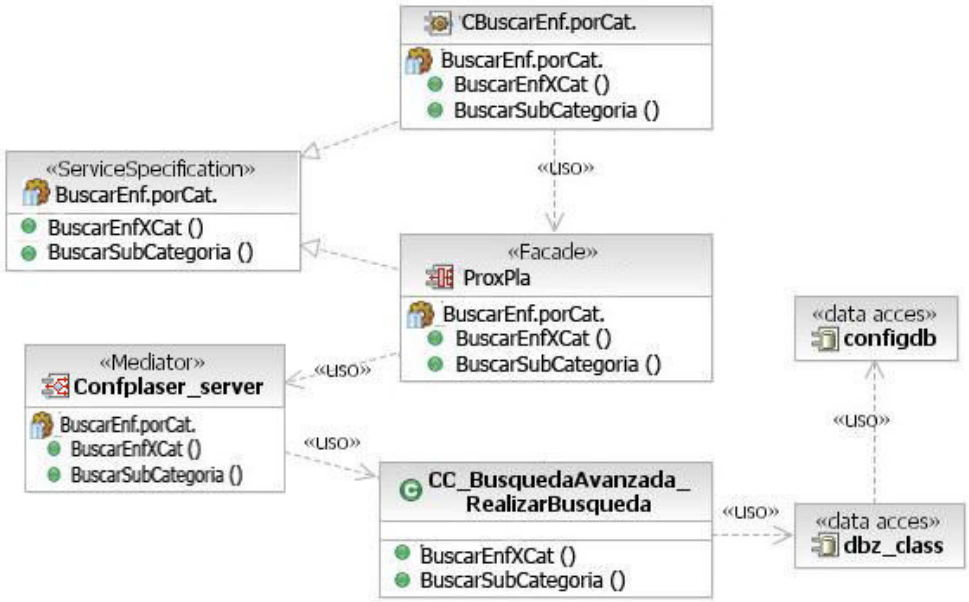
Especificación de Servicio: Buscar Enfermedad CIE
Figura 3.18 Realización de la especificación de servicio Buscar Enfermedad CIE



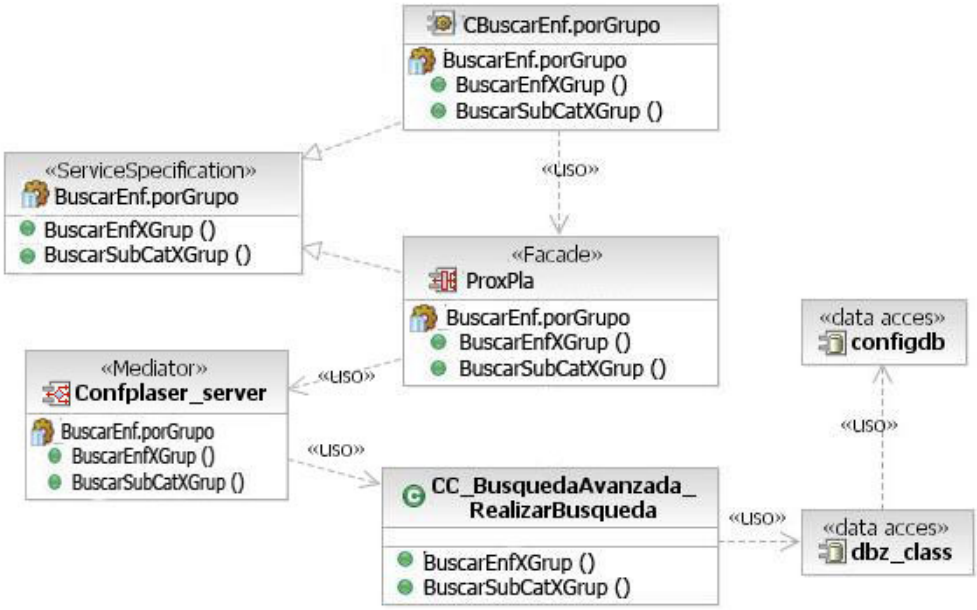
Especificación de Servicio: Buscar Enfermedad por Capítulo
Figura 3.19 Realización de la especificación de servicio Buscar Enfermedad por Capítulo



Especificación de Servicio: Buscar Enfermedad por Categoría
Figura 3.20 Realización de la especificación de servicio Buscar Enfermedad por Categoría.

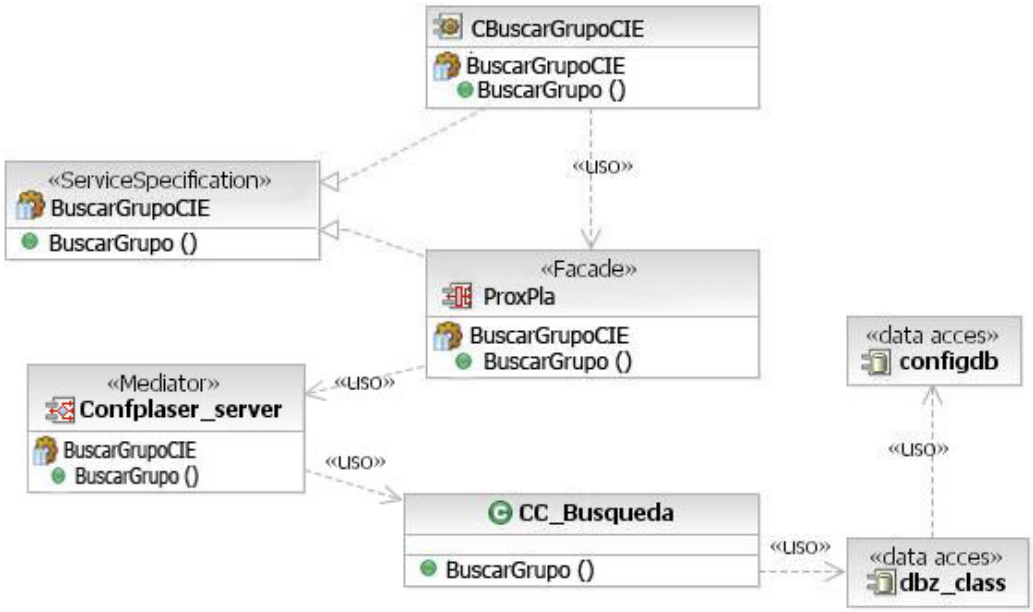


Especificación de Servicio: Buscar Enfermedad por Grupo
Figura 3.21 Realización de la especificación de servicio Buscar Enfermedad por Grupo.



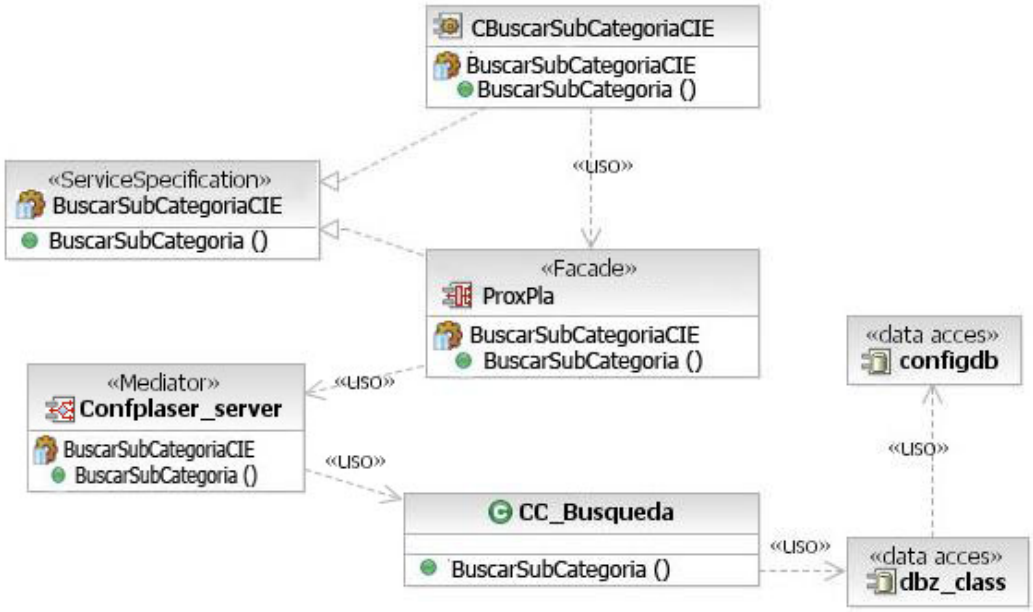
Especificación de Servicio: Buscar Grupo CIE

Figura 3.22 Realización de la especificación de servicio Buscar Grupo CIE.



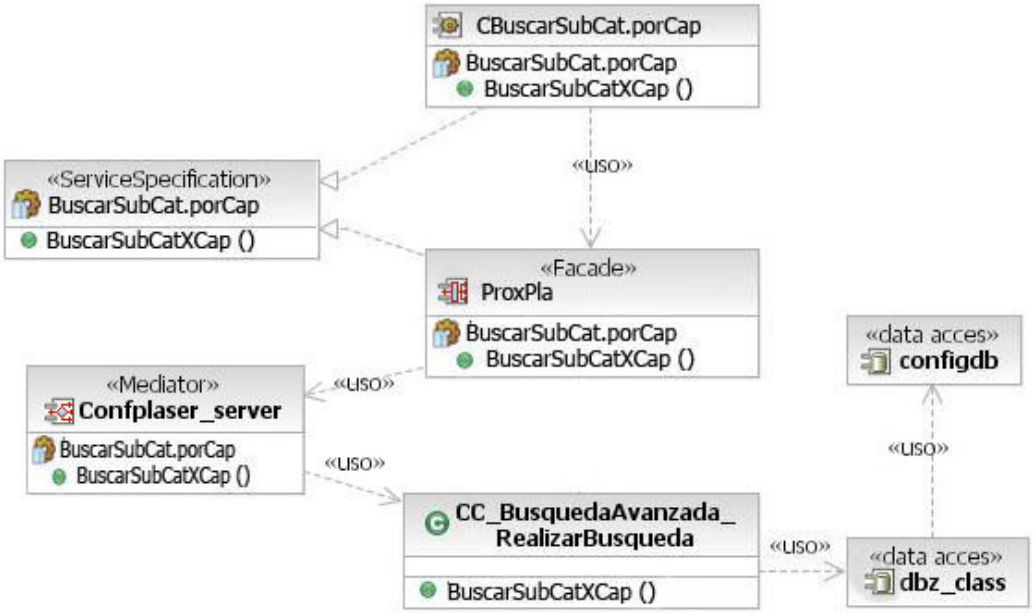
Especificación de Servicio: Buscar SubCategoría CIE

Figura 3.23 Realización de la especificación de servicio Buscar SubCategoría CIE.



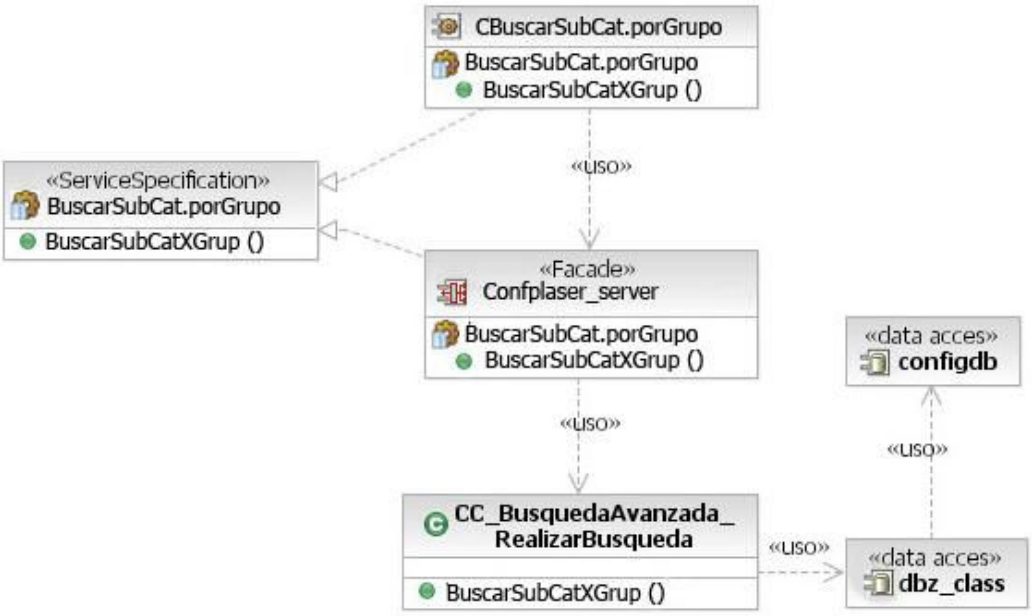
Especificación de Servicio: Buscar SubCategoría por Capítulo

Figura 3.24 Realización de la especificación de servicio Buscar SubCategoría por Capítulo.

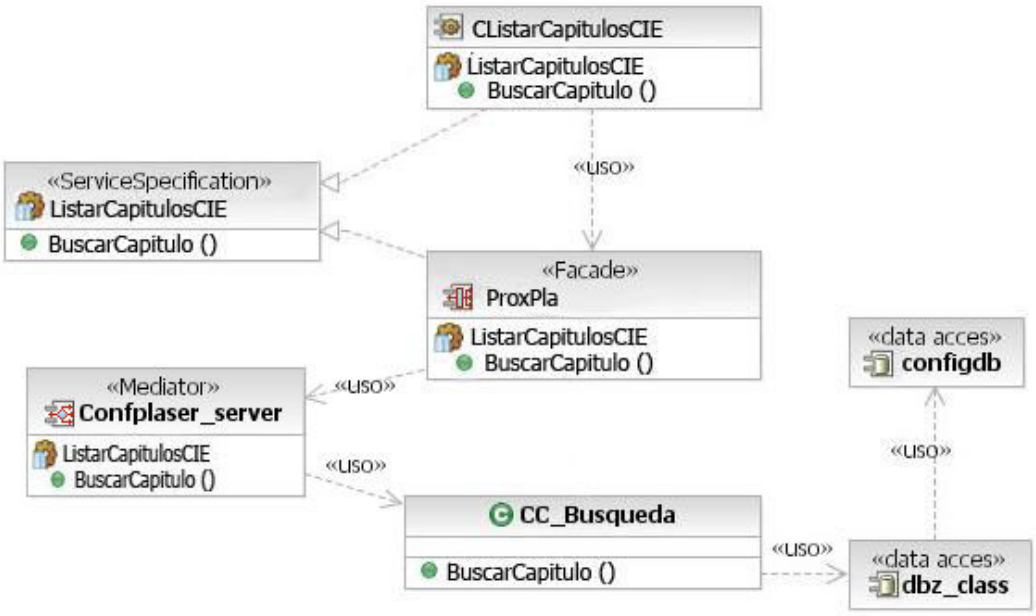


Especificación de Servicio: Buscar SubCategoría por Grupo

Figura 3.25 Realización de la especificación de servicio Buscar SubCategoría por Grupo.



Especificación de Servicio: Listar Capítulos CIE
Figura 3.26 Realización de la especificación de servicio Buscar Listar Capítulos CIE.



3.3 Modelo de Servicios de RCIE relacionado con SAAA.

Identificación de Servicios

Teniendo en cuenta los pasos a seguir para la realización de esta actividad se identificó el siguiente servicio:

- ✓ Autenticarse

Identificar Particiones de Servicios

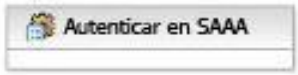
Debido a que solamente se tiene un servicio a realizar se considera que el mismo no se debe representar en una partición de servicios.

Desarrollo de Especificación de Servicios

Considerando el servicio identificado anteriormente y teniendo en cuenta que el consumo de este en RCIE con SAAA es a partir del fichero login.php que tiene el sistema SISalud (mediante el cual se accede a los diferentes módulos, entre ellos RCIE), se diseñó la especificación de servicios determinada para el método definido.

El siguiente diagrama muestra un árbol vacío de las especificaciones de servicios creadas en esta fase:

Figura 3.27 Especificación de Servicios Autenticar en SAAA



Diseño de Servicios

A continuación se representan los proveedores que en este caso es SAAA, relacionado con la especificación de servicios antes descrita y sus funciones. Resultando de la siguiente forma:

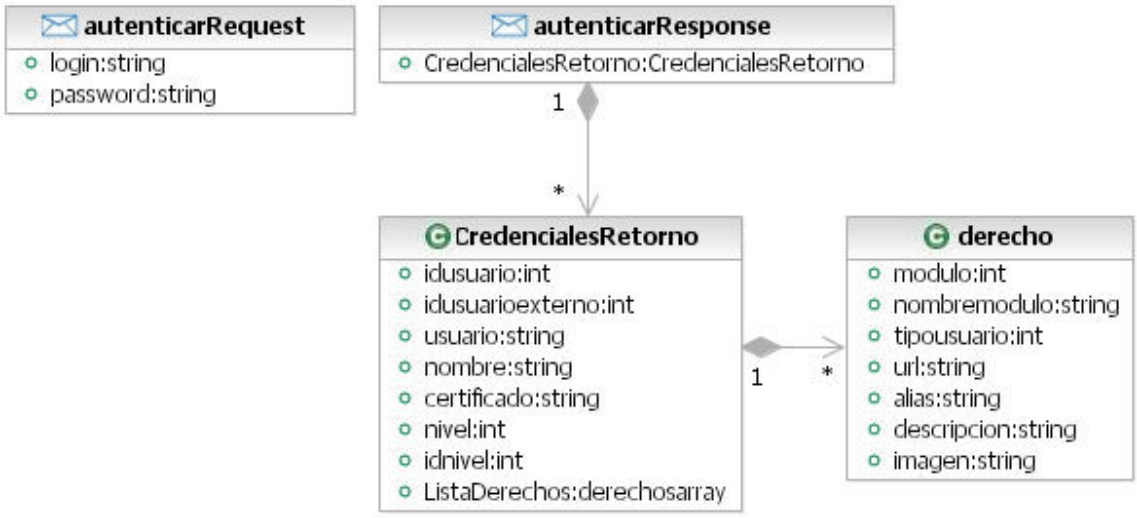
Figura 3.28 Diseño de Servicios

Diseño de mensajes

Para el diseño de mensajes a realizar se tuvo en cuenta que ya los mensajes están implementados, pero los mismos tienen cierta relación con el modelo de dominio descrito para SAAA y a su vez con las operaciones de las especificaciones de servicios antes descritas. En general para la especificación de servicios en SAAA existen dos ¨ tipos¨ de mensajes: Request (que contiene los datos introducidos en la petición del servicio) y Response (que contiene los datos que serán entregados).

Por otra parte, es bueno esclarecer que para la representación del diseño de mensajes, los mismos se obtuvieron a partir del Wsdl de este módulo, detallados aquí de una no forma lineal (pues los mensajes tienen una estructura compleja), por lo que queda configurado el modelo de diseño de la siguiente forma por el servicio identificado.

Figura 3.29 Diseño de Mensajes del Servicio Autenticar en SAAA



Realización de Servicios

Durante esta actividad se realiza un diagrama de la especificación de servicio detallada anteriormente. Para la realización de la especificación se siguieron los siguientes pasos:

Por cada especificación de servicio se debe tener un componente de servicios y un componente fachada relacionados a ella.

- ✓ También se deben representar los componentes de conexión a la base de datos.
- ✓ Las relaciones que deben existir entre cada elemento son de uso.
- ✓ Se deben representar los métodos del negocio del proveedor (SAAA) que responden a la implementación del servicio a tratar.

De forma general el desarrollo de cada especificación de servicios establecida, de SAAA para RCIE, tienen el siguiente comportamiento:

- ✓ Por cada especificación se generaliza un componente de servicios y un componente fachada.
- ✓ Teniendo en cuenta las funcionalidades del componente fachada que plantea el Plug-in se llegó a la conclusión que el componente Proxpla cumple con estos requisitos; este componente se encarga de recibir todas las peticiones que hace el sistema y de dar respuestas a las mismas. Configura los métodos empaquetados en un negocio con su tipo, nombre y especificaciones de acceso. Cuando RCIE hace una petición a SAAA esta petición primero llega al Proxpla, el cual verifica el método y el módulo abastecedor,

comportándose así como la primera capa de seguridad para luego dirigirse a la segunda que sería la clase Confplaser_server de SAAA, comportándose esta última como el mediador según los requisitos que encierra: este se usa para una o más aplicaciones para un funcionamiento de servicio dado, el mediador se llama por la fachada para identificar y llamar el componente de aplicación correcto.

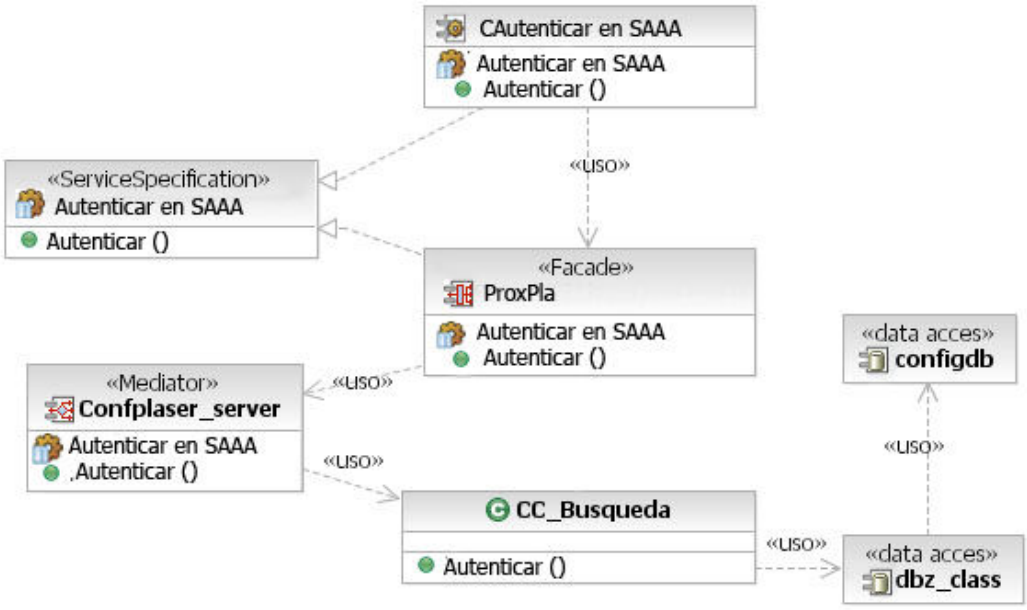
✓ Dado que el componente ConfPlaser_server contiene al método de SAAA que se quiere consumir, esta clase lo llama, y a su vez el método en su implementación utiliza a la clase dbz_class para la conexión a la base de datos, considerada esta clase como un componente de acceso a dato.

✓ La clase dbz_class implementa el acceso a cualquier tipo de Base de Datos. Dbz utiliza a ConfigDB para obtener la localización de la base de datos y los datos del usuario con derechos de inserción, asumiéndolo así como otro componente de acceso a datos.

✓ En este caso se representó la relación entre la clase mediador con la clase "ficticia" diseñada en el modelo de análisis y diseño, pero se debe tomar en cuenta que lo que existe son métodos dentro de ficheros (en este caso autenticar.php) los que son llamados por ConfPlaser_server.

Asumiendo lo antes expuesto a continuación se representan en el siguiente diagrama la realización de la especificación de servicios identificadas para representar el consumo de RCIE a SAAA:

Figura 3.30 Realización de la especificación de Servicios Autenticar en SAAA



3.4 Conclusiones.

La realización del modelo de servicios es un proceso que puede ser tan complejo como la construcción del modelo del análisis y diseño. Es bueno considerar que este no se divorcia del último ya que guardan una estrecha relación. En este capítulo se trató de llegar a un acercamiento en ese aspecto, para así dar solución, tomando como guía al Plug-in SOA-RUP, a la representación de la transferencia de servicios en RCIE, tanto como componente consumidor como proveedor.

CONCLUSIONES

CONCLUSIONES

Con la realización del presente trabajo se llegaron a cumplir los siguientes aspectos dándole respuesta a las tareas trazadas:

✓ Se examinó el uso del Plug-in SOA-RUP para el Rational profundizando en la metodología los artefactos y modelos a realizar relacionados con los servicios, pudiéndose determinar como los más significativos:

- Diseño del Mensaje
- Diseño de Servicios
- Modelo de Servicios
- Artefactos del modelo de servicios.
- Modelo de diseño

✓ Se valoró el modelo de análisis y diseño que se realizó para el desarrollo de RCIE, detectándose las áreas menos desarrolladas con respecto a SOA, y a partir de ahí se formalizaron sus especificidades con el Plug-in SOA-RUP para la creación de un modelo de servicio.

Con la ejecución de todas estas actividades se logró dar respuesta al objetivo general del trabajo de realizar los cambios necesarios en la ingeniería del software del módulo RCIE basándose en el Plug-in SOA-RUP implementado para el Rational Unified Process Builder, de forma tal que se adaptará a una arquitectura orientada a servicios.

RECOMENDACIONES

RECOMENDACIONES

En sentido general, se puede establecer que la utilización del Plug-in SOA-RUP, proporcionará una serie de beneficios para el desarrollo de aplicaciones Web basadas en arquitectura orientada a servicios, pudiéndose adaptar perfectamente a las necesidades y características del proyecto APS en general. Para su puesta en práctica se hacen las siguientes recomendaciones:

- ✓ Mantenerse actualizado en cuanto a nuevas versiones del Plug-in que puedan aportar informaciones novedosas.
- ✓ Extender la metodología SOA-RUP propuesta, a todos los módulos de APS en fase de inicio o en desarrollo.
- ✓ Usar la herramienta Rational Software Architect (RSA) recomendada para este proceso de ingeniería por el Plug-in. Para ello se debe tener en cuenta obtener la versión correcta que contenga como plantilla de proyecto UML, el Modelo de Diseños de Servicios.
- ✓ Aplicar el trabajo del RSA en el proyecto APS, para realizar no solo los diagramas del modelo de servicios sino los demás que plantea RUP, y de esta forma, obtener un modelo general en un solo archivo.
- ✓ Diseñar en el RSA una vista global de la arquitectura SOA del proyecto APS.

Otras recomendaciones muy importantes serían:

- ✓ Analizar como otra alternativa la Plataforma Websphere como infraestructura para la Integración, aplicación y una posible utilización en APS.
- ✓ Estudiar el proceso de integración SOA/BPM.

BIBLIOGRAFÍA Y REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

1. Guillarte, O. C. Registro de Población para el Sistema Informatizado de la Atención Primaria (RPOB). Facultad de Ingeniería Industrial. Ciudad Habana, Instituto Superior Politécnico “José Antonio Echeverría”, 2005.p.
2. Ídem 1
3. Ídem 1
4. SÁNCHEZ, J. M. Infraestructura de Desarrollo de Software. Ciudad Habana, Empresa Softel, 2006.
5. Ídem 4
6. Cabrera Hernández, Mirna et al. Propuesta de Esquema del Sistema de Información para la Salud (SISalud). La Habana. Cuba. 2006.
7. Ídem 4
8. Ídem 4
9. Ídem 4
10. Ídem 4
11. Ídem 4
12. Ídem 4
13. SÁNCHEZ, J. M. Descripción de la Arquitectura. Ciudad Habana, Empresa Softel, 2006.
14. MEHDI. SOA requirements & service use cases IBM Rational, 2005. [Disponible en: http://www.128.ibm.com/developerworks/forums/dw_thread.jsp?forum=335&thread=143496&cat=67.
15. TOMASEVICH, M. IBM RUP for Service-Oriented Modeling and Architecture V2.4 now available on developerWorks!, IBM Rational, 2006. [2007]. Disponible en: http://www.128.ibm.com/developerworks/forums/dw_thread.jsp?message=13893490&cat=24&thread=143905&treeDisplayType=threadmode1&forum=335#13893490
16. Ídem 1
17. Ídem 1

18. AUTORES, C. D. Documento sobre la Arquitectura de Software para los componentes a emplear por el Sistema de Información para la Salud. Ciudad Habana, Empresa Softel, 2006.
19. Ídem 18
20. SOLANO, P. Infome Especial. SOA Arquitectura Orientada a Servicios., [Informe Especial]. GBM, 2005. [Disponible en: http://www.gbm.net/bluetech/BT31/10_bluetech.html.
21. Ídem 20
22. Ídem 20
23. Ídem 20
24. Ídem 20
25. Ídem 20
26. Ídem 20
27. Ídem 20
28. Ídem 20
29. Ídem 20
30. BRAVO, S. P. Arquitectura SOA e Integración de aplicaciones., Autentia, 2005. [2006]. Disponible en: http://www.autentia.com.
31. Ídem 1
32. Ídem 18
33. MARIN, J. Documentación del Proyecto. Ciudad Habana, Empresa Softel, 2005. 2006.
34. Ídem 34
35. Ídem 34
36. Ídem 34
37. Idem34
38. Idem34
39. Ídem 34
40. Ídem 34
41. Ídem 34
42. Ídem 34
43. Ídem 34
44. Ídem 34

45. Ídem 34
46. Ídem 34
47. Jacobson, Ivar; Booch, Grady; Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. La Habana. Cuba. Editorial Félix Varela. 2004. Pág. 4, 5, 6 y 7.
48. Ídem 48
49. Ídem 1
50. Ídem 1
51. Ídem 1
52. Ídem 1
53. Ídem 1
54. Ídem 1
55. Ídem 13

BIBLIOGRAFÍA

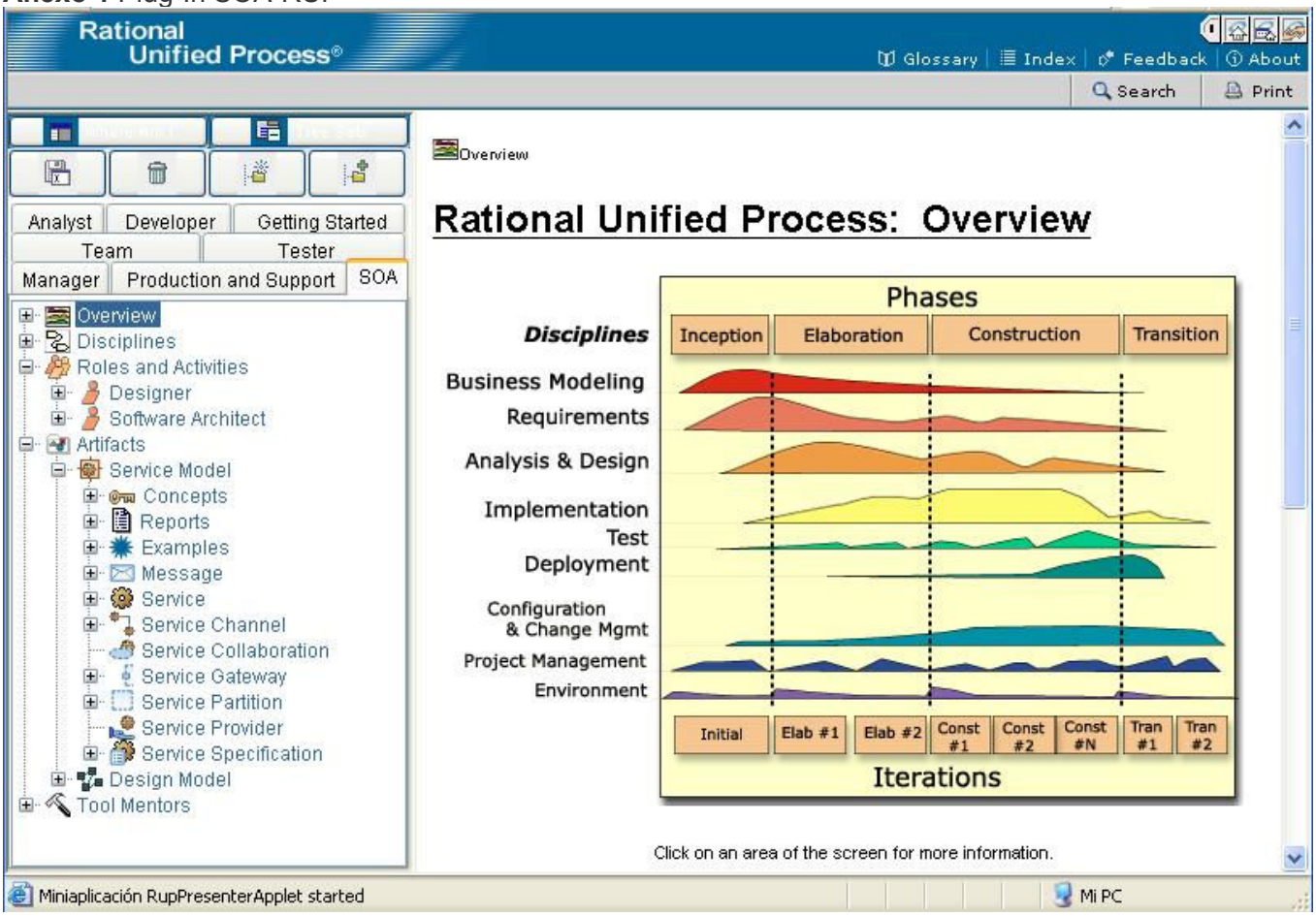
- ALTMAN, R. SOA en la convergencia, Bloggers Report, 2006.
- AUTORES, C. D. Documento sobre la Arquitectura de Software para los componentes a emplear por el Sistema de Información para la Salud. Ciudad Habana, Empresa Softel, 2006.
- AUTORES, C. D. "Patrones Arquitectónicos en las Aplicaciones de Atención Primaria de la Salud (APS)", 2006.
- BRAVO, S. P. Arquitectura SOA e Integración de aplicaciones., Autentia, 2005. [2006]. Disponible en: http://www.autentia.com.
- Cabrera Hernández, Mirna et al. Propuesta de Esquema del Sistema de Información para la Salud (SISalud). La Habana. Cuba. 2006.
- CELAYA, C. L. Arquitectura de Servicios WEB, 2006. [Disponible en: http://glud.udistrital.edu.co/glud/areas/doc/articulos/1_articulo_ws/servicios-web.html.
- DESBOUIS, T. BPM y SOA, herramientas complementarias para empresas ágiles, BAQUIA, 2007. [Disponible en: http://www.baquia.com/noticias.php?id=11894
- Guillarte, O. C. Registro de Población para el Sistema Informatizado de la Atención Primaria (RPOB). Facultad de Ingeniería Industrial. Ciudad Habana, Instituto Superior Politécnico "José Antonio Echeverría", 2005.p.
- Jacobson, Ivar; Booch, Grady; Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. La Habana. Cuba. Editorial Félix Varela. 2004.
- MARIN, J. Documentación del Proyecto. Ciudad Habana, Empresa Softel, 2005. 2006.
- MEHDI. SOA requirements & service use cases IBM Rational, 2005. [Disponible en:http://www.128.ibm.com/developerworks/forums/dw_thread.jsp?forum=335&thread=143496&cat=67.
- MILLER, C. 7 razones por las que SOA hará tambalear su mundo 2006. [Disponible en: http://www.help400.es/asp/scripts/nwart.asp?Num=167&Pag=10&Tip=T.
- PARRA, J. D. Hacia una Arquitectura Empresarial basada en Servicios MSDN, 2006. [2007]. Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp
- SÁNCHEZ, J. M. Infraestructura de Desarrollo de Software. Ciudad Habana, Empresa Softel, 2006.

- SÁNCHEZ, J. M. Descripción de la Arquitectura. Ciudad Habana, Empresa Softel, 2006.
- SOLANO, P. Informe Especial. SOA Arquitectura Orientada a Servicios., [Informe Especial]. GBM, 2005. [Disponible en: http://www.gbm.net/bluetech/BT31/10_bluetech.html.
- TOMASEVICH, M. IBM RUP for Service-Oriented Modeling and Architecture V2.4 now available on developerWorks!, IBM Rational, 2006. [2007]. Disponible en: http://www128.ibm.com/developerworks/forums/dw_thread.jsp?message=13893490&cat=24&thread=143905&treeDisplayType=threadmode1&forum=335#13893490
- WIKIPEDIA. Arquitectura orientada a servicios, Wikimedia Foundation, Inc., 2007. [Disponible en: http://es.wikipedia.org/wiki/SOA

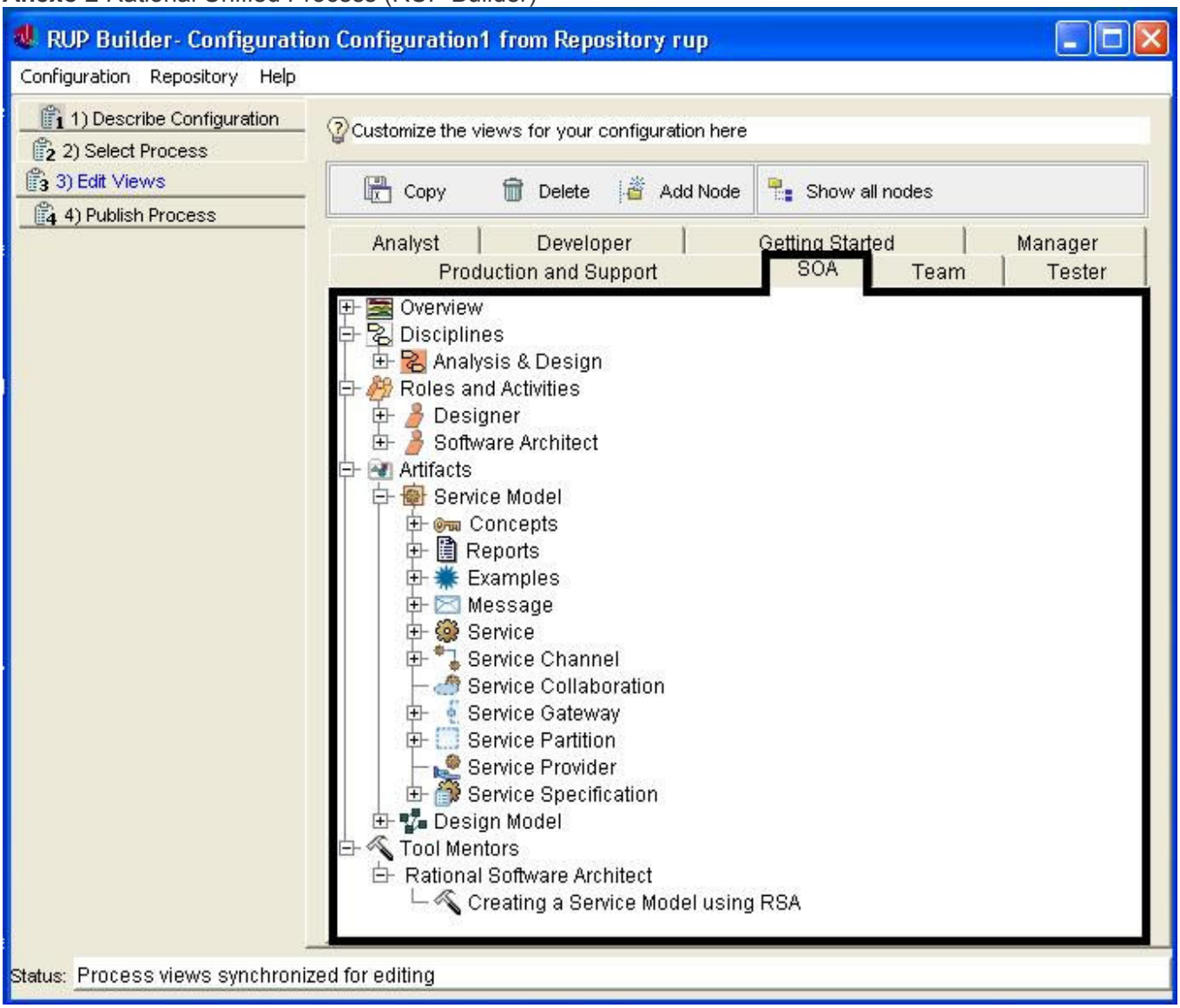
ANEXOS

ANEXOS

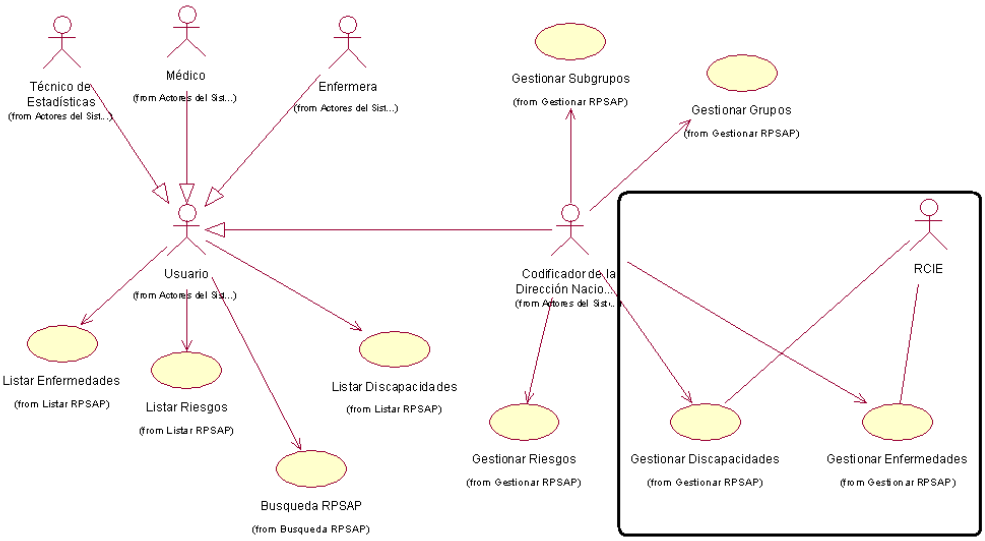
Anexo 1 Plug-In SOA-RUP



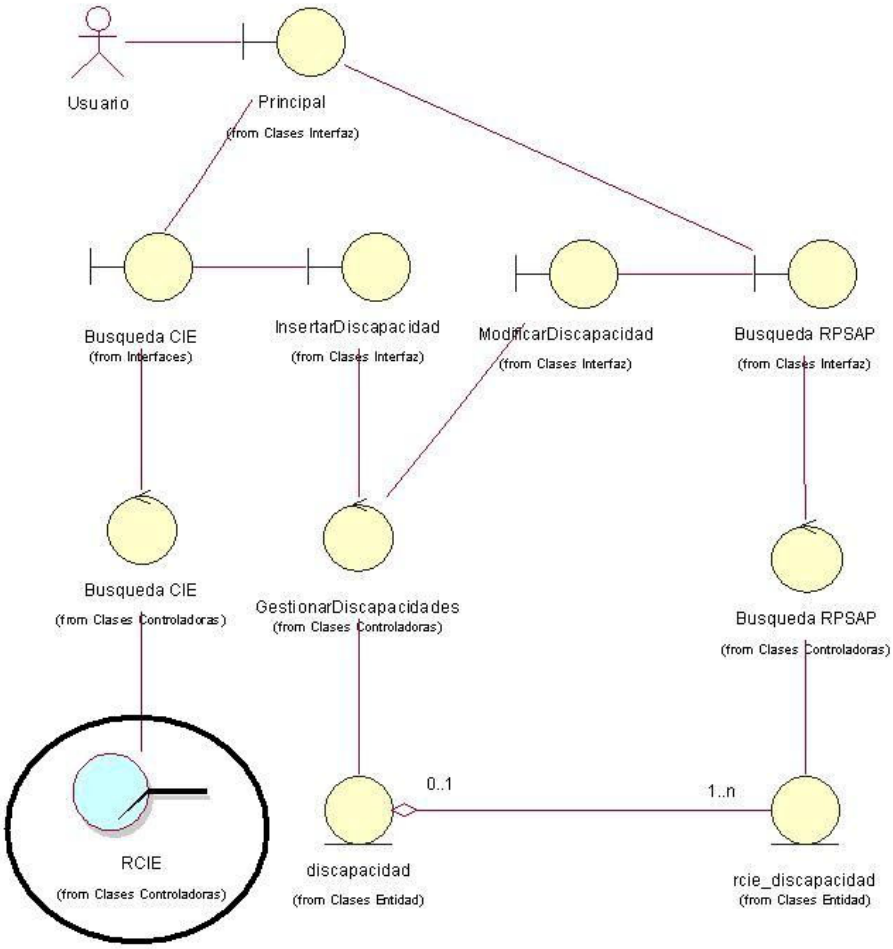
Anexo 2 Rational Unified Process (RUP Builder)



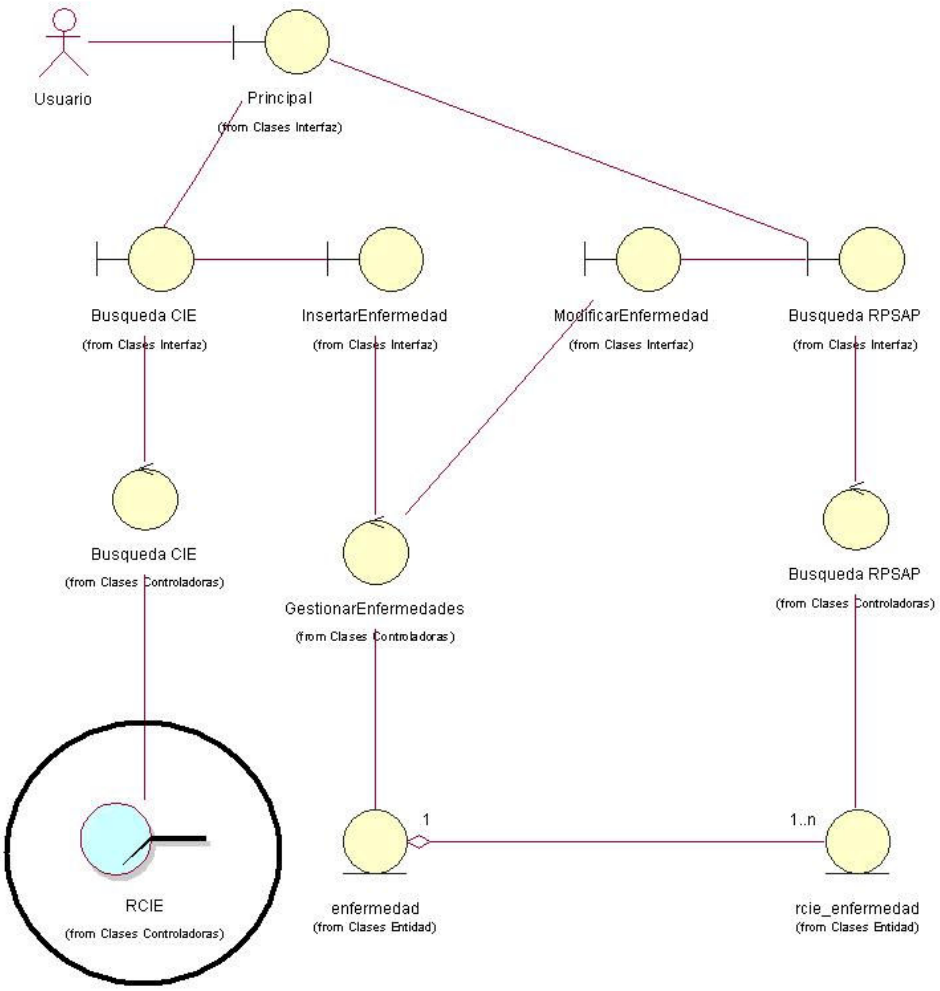
Anexo 3 Diagrama de Casos de Uso del Sistema de RPSAP



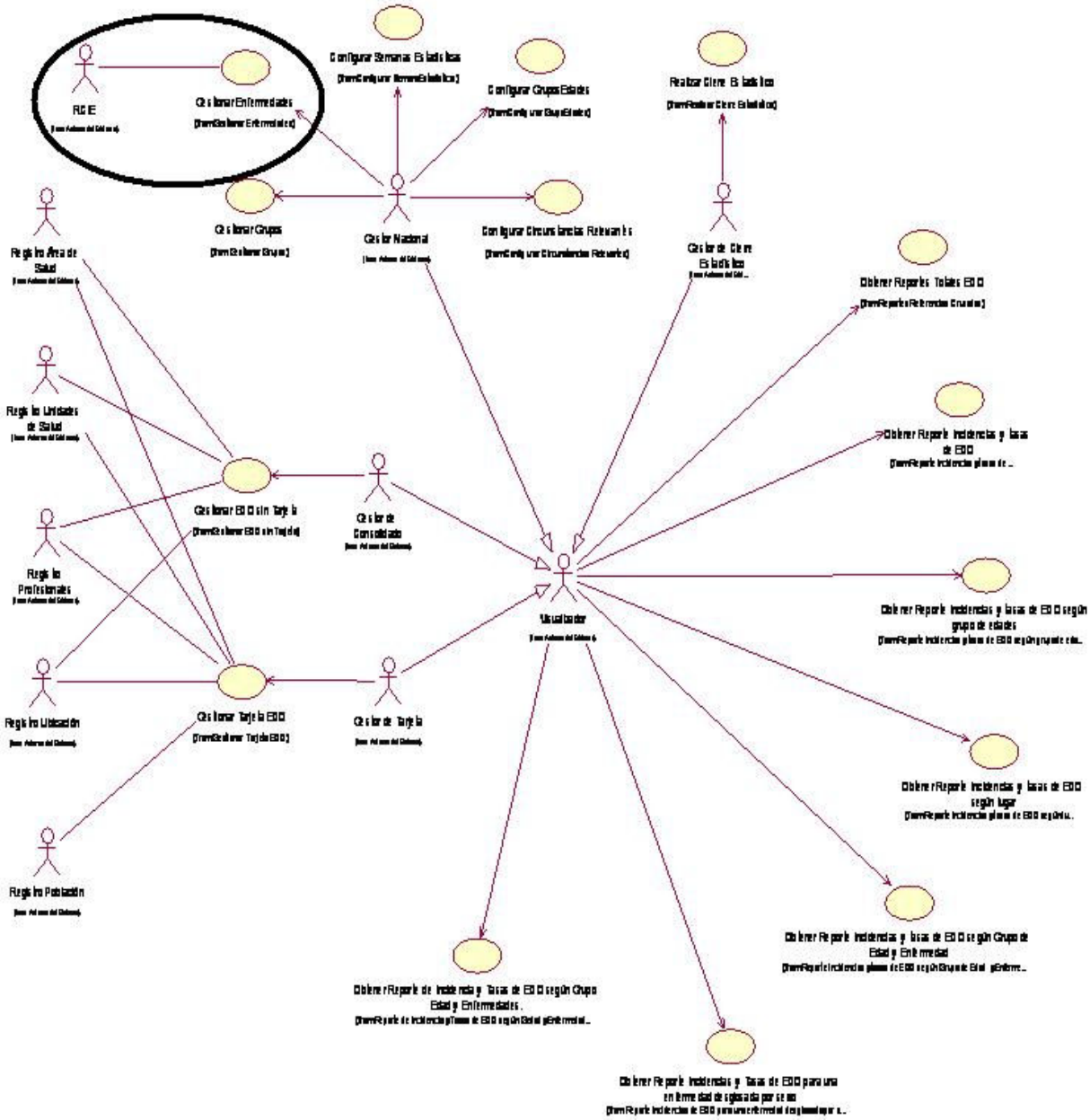
Anexo 4 Diagrama de Clases del Análisis de RPSAP (Caso de uso Gestionar Discapacidades)



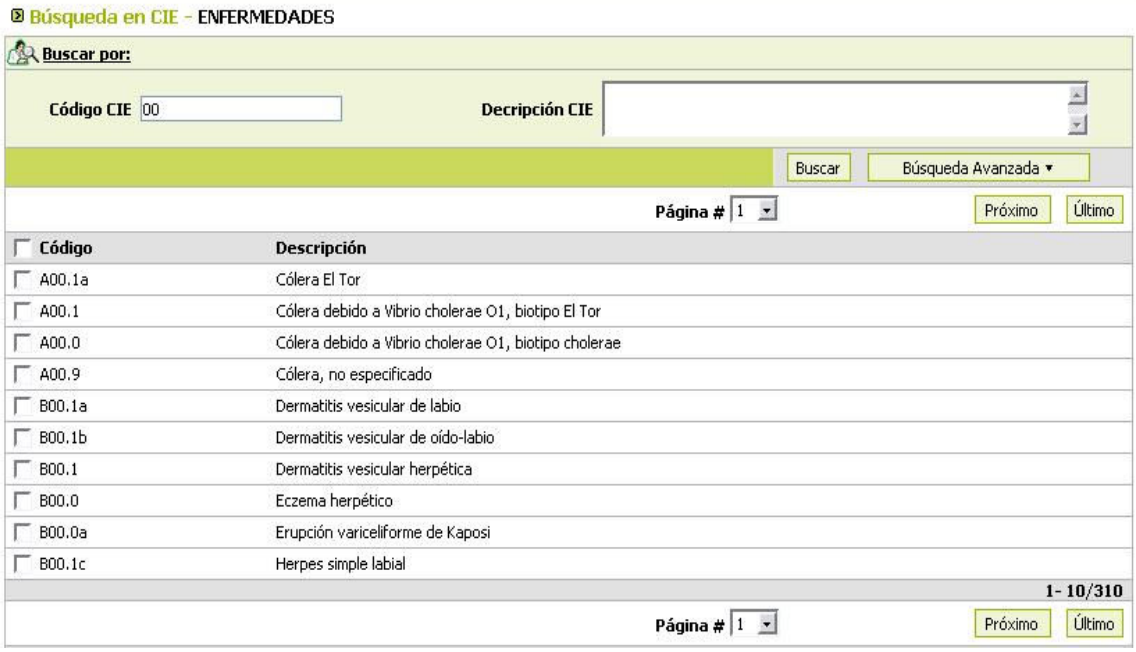
Anexo 5 Diagrama de Clases del Análisis de RPSAP (Caso de Uso Gestionar Enfermedades)



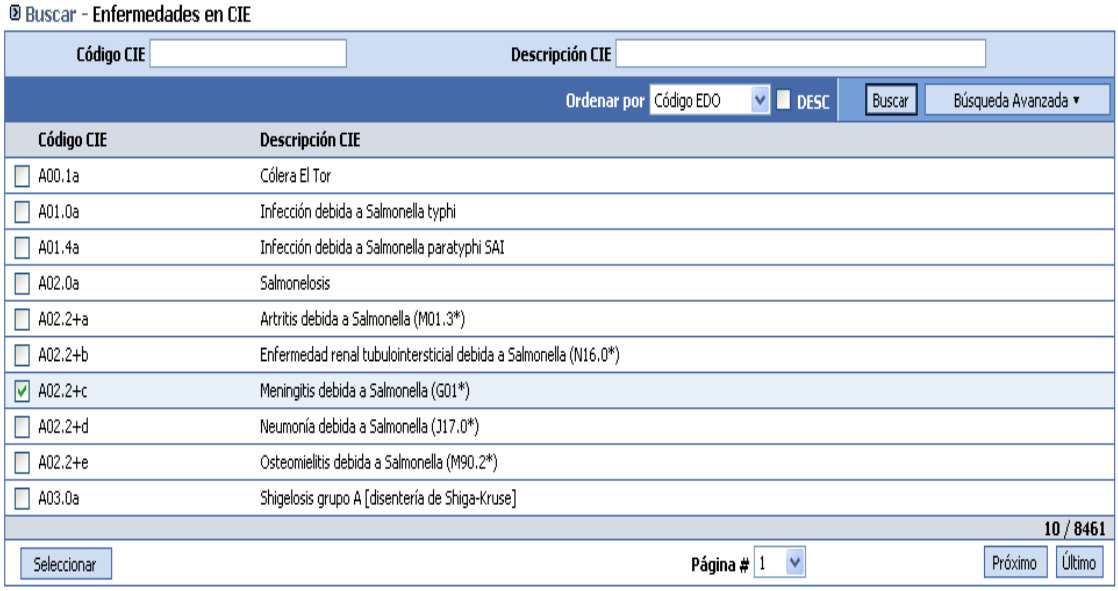
Anexo 6 Diagrama de Casos de Uso del Sistema de REDO



Anexo 7 Aplicación RPSAP (consumidor de RCIE)



Anexo 8 Aplicación REDO (consumidor de RCIE)



Anexo 9 Acceso a SISalud

GLOSARIO DE TÉRMINOS

GLOSARIO DE TÉRMINOS

1. Artefactos: son los productos tangibles del proceso unificado de desarrollo de software. Son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables, documentos, etc.

2. Estereotipos: El concepto inicial de estereotipo (*stereotype*) se refiere a una clasificación de alto nivel de un objeto que proporciona una idea del tipo de objeto del que se trata. Dicho concepto ha sido recogido en UML para proporcionar un mecanismo de extensión para el propio lenguaje y de esta forma proporcionar artefactos de modelado que tienen su significado dentro de un proceso *software* específico. Un estereotipo será un nuevo tipo de elemento de modelado que extiende la semántica del metamodelo pero no la estructura de los tipos o clases preexistentes. Gráficamente un estereotipo se representa como un nombre entre comillas <<>>.

3. File System: es un sistema de archivos distribuido a través de la red que fue desarrollado como parte del proyecto Andrew por parte de la Universidad Carnegie Mellon. Su nombre proviene de Andrew Carnegie y Andrew Mellon. Su uso fundamental está en la computación distribuida.

4. Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Los Frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

5. Herramientas Cases: Las **Herramientas CASE** (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

6. IDL: Interface Definition Language (IDL) es, como su propio nombre indica un lenguaje de especificación de interfaces que se usa como parte de la tecnología CORBA. Ofrece la sintaxis necesaria para definir los métodos que queremos invocar remotamente.

7. J2EE: Java Platform, Enterprise Edition o Java EE es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de n niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad.

8. Object Management Group: El Object Management Group u OMG (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.

9. Parser: Un analizador sintáctico (parser) en informática y lingüística es un proceso que analiza secuencias de tokens para determinar su estructura gramatical respecto a una gramática formal dada. Un parser es así mismo un programa que reconoce si una o varias cadenas de caracteres forman parte de un determinado lenguaje, es utilizado por ejemplo en compiladores.

10. Plug in: Un **Plug-in** (o **Plug-in** -en inglés "enchufar) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica. Ésta aplicación adicional es ejecutada por la aplicación principal.

11. SOA/BPM: Disciplina empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio (BPR), que se deben modelar, automatizar, integrar, monitorizar y optimizar de forma continua.Como su nombre lo sugiere Business Process Management (BPM) se enfoca en la administración de los procesos del negocio. A través del modelado de las actividades y procesos se logra un mejor entendimiento del negocio y muchas veces esto presenta la oportunidad de mejorarlos. La automatización de los procesos reduce errores, asegurando que los mismos se comporten siempre de la misma manera y dando elementos que permitan visualizar el estado de los mismos. La administración de los procesos permite asegurar que los mismos estén ejecutándose eficientemente y obtener información

que luego puede ser usada para mejorarlos. Es a través de la información que se obtiene de la ejecución diaria de los procesos que se puede identificar posibles ineficiencias en los mismos y de esta forma optimizarlos. Para soportar esta estrategia es necesario contar con un conjunto de herramientas que den el soporte necesario para cumplir con el ciclo de vida de BPM. Este conjunto de herramientas son llamadas Business Process Management System y con ellas se construyen aplicaciones BPM.

12. SGML: Son las siglas de "Standard Generalized Markup Language" o "Lenguaje de Marcación Generalizado". Consiste en un sistema para la organización y etiquetado de documentos. El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial. El lenguaje HTML esta definido en términos del SGML.

13. Socket: designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

14. Software AG: fundada en 1969 en Darmstadt, Alemania, es actualmente líder mundial en soluciones y tecnología basadas en XML, estándar clave para la integración y desarrollo de negocios.

15. SQL ANSI 92: estándar de SQL (Lenguaje de Consulta Estructurado para Base de Datos), es general y amplio.

16. Sun Microsystems: es una empresa informática del Silicon Valley, fabricante de semiconductores y software. Las siglas SUN se derivan de «**S**tanford **U**niversity **N**etwork». En ese año introducen al mercado su primera estación de trabajo que desde su inicio trabajó con el protocolo TCP/IP.

17. Tags: (o etiqueta) Es una marca con tipo que delimita una región en los lenguajes basados en XML. Por ejemplo: <etiqueta1> <etiqueta2 atributo1="hola" atributo2="mundo"> </etiqueta2> <etiqueta3 atributo1="domin"/> </etiqueta1>.

18. Visual Paradigm: Herramienta CASE que da soporte al modelado visual con el lenguaje de modelado UML 2.0. Está disponible en varias ediciones, cada una destinada a varias necesidades: de Empresa, Profesional, de Comunidad, Estándar, de Modelamiento y Personal. Tiene una versión gratuita.

19. Websphere: La plataforma de WebSphere está basada completamente sobre los principios ya expuestos de las Arquitecturas Orientadas a Servicios. Todos los productos de IBM y particularmente los de la familia de productos de WebSphere soportan la integración a través de Web Services y brindan las características necesarias para implementar una Arquitectura Orientada a Servicios que comprenda tanto los sistemas existentes como nuevos desarrollos sobre tecnologías de punta como J2EE. WebSphere ha sido calificado por muchos analistas internacionales como la plataforma líder para implementar SOA. Entre los elementos más conocidos de la familia de productos de WebSphere está el WebSphere Enterprise Service Bus (ESB), que es la plataforma que brinda los servicios de enrutamiento y transformación de mensajería para la arquitectura SOA basado en XML.

20. XSLFO: Un documento XSL-FO es un documento XML en el que se especifica cómo se van a formatear unos datos para presentarlos en pantalla, papel u otros medios. El significado de las siglas XSL-FO es e**X**tensible **S**tylesheet **L**anguage **F**ormatting **O**bjects. Hay que destacar que en el documento XSL-FO figuran tanto los datos como el formato que se les va a aplicar.