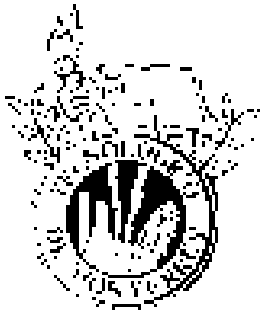
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS MONTERREY

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES



Propuesta de un proceso para desarrollar Web Services utilizando UML

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADÉMICO DE:

MAESTRO EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR:

MARTHA JANET DE LA LUZ GARCÍA LÓPEZ

MONTERREY, NUEVO LEÓN **MAYO 2003**

Propuesta de un Proceso para Desarrollar Web Services Utilizando UML

POR:
MARTHA JANET DE LA LUZ GARCÍA LÓPEZ

TESIS

Presentada a la División de Graduados en Computación, Información y Comunicaciones

Este trabajo es requisito parcial para obtener el título de
Maestro en Ciencias en Tecnología Informática

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

MAYO 2003

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

DIVISIÓN DE ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

Los miembros del comité de tesis, recomendamos que esta tesis presentada por Martha Janet de la Luz García López sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias, especialidad en

Tecnología Informática

Comité de tesis:

_____

Dr. Guillermo Jiménez Pérez
Asesor Principal

_____

Dr. Juan Carlos Lavariega Jarquín
Sinodal

_____

Dr. José Raúl Pérez Cázares
Sinodal

_____

Dr. David A. Garza Salazar
Director de los Programas de Postgrado en Computación, Información y Comunicaciones

MAYO 2003

Dedicatoria

A Dios, por guiarme en su camino, hacia la verdad, la luz y la vida.
“dame señor entendimiento sereno, para poder resolver las cosas con dulce calma”

A mi madre, ejemplo de tesón y fortaleza, por enseñarme a luchar por un ideal...
como siempre por su infinito amor.

Reconocimientos

A mis padres, Gregorio y Martha por su apoyo, cariño y sus palabras de aliento. Para mi madre en especial, por sus rezos, por sus horas de desvelo, por estar conmigo y apoyarme en todo momento aún sin estar de acuerdo. A mis hermanos, Olga, Olivia, Sandra, Eduardo, Josué por reservarme un lugar en casa a pesar de la distancia.

A mi asesor Dr. Guillermo Jiménez Pérez por su orientación principalmente, por su tiempo y ayuda.

Al Dr. José Raúl Pérez Cázares y al Dr. Juan Carlos Lavariega Jarquín por sus comentarios y participación como sinodales en este proyecto de investigación.

Al resto de mi familia, en especial a mi tío Armando por su apoyo.

A mis amigos tec...
a mi equipo: Ido y Ramón, por todo el apoyo, sobretodo por los últimos días. A Raúl por su sincera amistad. A Jorge por su colaboración. A Perla, por escuchar, por todo lo compartido, en especial por su apoyo incondicional y motivación.

A mis amigos de siempre...
A ti Raúl por tus sabios consejos, por todas tus terapias y por seguir conociéndome a miles de kilómetros... porque sigues aquí. A ti Ricardo por tus loqueras, por tu apoyo y por tu ejemplo para ver la vida con frialdad y ligereza. A Adriana, Oste, Laura, Araceli, Ramiro, Clau, Betty, Fer porque sé que disfrutan la realización de esta meta tanto como yo.

A Vikas, my battery, por todos estos años, por su cariño imponderable, por todas esas palabras en el momento preciso, lu.

A todos mis compañeros de MCT por el tiempo compartido y por el esfuerzo realizado.

A todas las personas que tuvieron una palabra de aliento y que de alguna manera contribuyeron en esta meta y que quizá ahora no los recuerde.

Resumen

Web Services se define como una interface para la integración de aplicaciones que describe una colección de operaciones que están accesibles en la red, es independiente de plataformas de desarrollo y basado en estándares abiertos. A la implementación del Web Services se le llama servicio, en otras palabras un servicio es un módulo de software. Los Web Services requieren para la compartición de servicios de una descripción del servicio que desean compartir, registrar los servicios para que puedan ser encontrados por otros Web Services y un medio de comunicación para el intercambio de servicios a través de mensajes.

En la investigación de este trabajo se identificó la siguiente clasificación de los requerimientos para Web Services: tecnológicos, de integración y orientados al servicio. Los Web Services al igual que cualquier otro proyecto de software requiere de un proceso de desarrollo.

El objetivo de este trabajo es proponer una serie de actividades para la realización de cada fase del desarrollo de Web Services, para lo cual fueron analizadas cuatro diferentes metodologías realizadas por Larman, C., Oestereich, B. y Rosenberg-Scott para el análisis y diseño orientado a objetos y Conallen, J., para aplicaciones Web. Del estudio realizado de éstas metodologías se obtuvo que ninguna era completamente aplicable para la elaboración de Web Services porque no satisfacen por completo los requerimientos.

Se propone un proceso iterativo que define la elaboración de dos documentos iniciales y una serie de actividades ordenadas y divididas en seis fases. Los documentos especifican una introducción y un panorama general. Las fases en las que se divide el proceso son: requerimientos, definición de arquitectura, análisis, diseño, codificación e implementación y pruebas, utilizando Unified Modeling Language (UML) como lenguaje de modelación.

Para la representación de los elementos de Web Services realizamos extensiones a UML para la modelación de las asociaciones entre clases para la representación de los estándares de Web Services y proponemos la utilización de componentes estereotipados para la representación de Web Services.

El proceso fue validado mediante dos casos de estudio en los que se obtuvo que los diagramas resultaron adecuados para la modelación de cada fase.

Índice

Índice de figuras

Índice de tablas

Índice de tablas

Capítulo 1 Introducción

1.1. Introducción

El desarrollo del Web trajo consigo una diversidad de aplicaciones, tales como el correo electrónico, comercio electrónico, inicialmente entre clientes con negocios y más recientemente negocios con negocios.

Web Services es la última generación en la evolución del Web a nivel integración de aplicaciones, independiente de plataformas y basada en estándares abiertos. Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI) y Web Services Description Language (WSDL) forman el conjunto de estándares que permiten descubrir, publicar, describir y utilizar Web Services.

Sin embargo el concepto Web Services esta todavía en etapa de desarrollo, incluso aún no existe una definición común. Pero partiendo de la premisa básica de que Web Services son interfaces disponibles en computadoras remotas, que usan un protocolo específico y se accedan vía Web principalmente, se derivan las siguientes interpretaciones: Services se refiere a brindar accesibilidad a un conjunto coherente de funcionalidad de la aplicaciones y Web se refiere a la forma como esa funcionalidad es accesada, por lo que podemos decir que un Web Services son servicios accesados a través de la red.

Como consecuencia del todavía incipiente desarrollo de Web Services encontramos que es una área con muchos rubros factibles de aportaciones. Donde una de éstas áreas de oportunidad, el hecho de que no se cuenta con un proceso definido, el cual es necesario para especificar la organización de las actividades del desarrollo de un sistema.

En este documento proponemos un proceso que sirva como referencia a los desarrolladores de Web Services, utilizando UML como lenguaje de modelación y siguiendo un proceso iterativo. Se especifican los diagramas que consideramos apropiados para describir cada una de las etapas del proceso propuesto y se sugieren extensiones a UML para el modelado de Web Services.

1.2. Objetivo

Proponer un proceso que sirva como guía en el desarrollo de Web Services.

1.3. Definición del problema

Actualmente se desarrollan Web Services, pero no se ha definido un proceso que apoye a los desarrolladores en cada una de las etapas. Los procesos de desarrollo tradicionales resultan inadecuados para Web Services [KJKS12001].

Encontramos entonces que los desarrolladores se enfrentan al problema de qué hacer, cómo hacer y cuándo hacer las actividades necesarias para desarrollar Web Services.

Lo que proponemos con el presente trabajo es identificar la secuencia ordenada de actividades y los artefactos para cada etapa del desarrollo de un proyecto de software que responda a las anteriores cuestiones. Basándonos en requerimientos tecnológicos, de integración y orientados al servicio que encontramos definen a los Web Services.

1.4. Motivación

La necesidad de integrar aplicaciones desarrolladas en diferentes plataformas esta incrementándose cada vez; Web Services está resultando adecuado para satisfacerla [BO2001]. El enfoque de Web Services comprende un grupo de estándares abiertos que facilitan la integración de aplicaciones con aplicaciones, lo que significa describir, publicar, descubrir y utilizar las interfaces de las aplicaciones. Tecnologías como UDDI, WSDL, SOAP y XML hacen posible que ésta integración sea posible sin requerir de una infraestructura compleja [GSBDDNN2002].

Web Services es relativamente nuevo, aún hay mucho que aportar en aspectos como arquitecturas, herramientas, seguridad, diseño. Nosotros vamos a trabajar el área de definir un proceso para desarrollar Web Services.

Nuestro objetivo es proveer a los desarrolladores de un proceso que especifique el conjunto de actividades y los artefactos para cada etapa. Este proceso estará basado en el análisis de cuatro diferentes propuestas de metodologías para el desarrollo de aplicaciones orientadas a objetos y de aplicaciones Web. Con la finalidad de identificar los puntos fuertes y débiles de cada propuesta que puedan ser relevantes para Web Services y proponer un de desarrollo orientado a Web Services, que tome los aspectos fuertes de cada propuesta o los complemente si es el caso.

Para modelar los artefactos de cada actividad se ha seleccionado Unified Modeling Language (UML), por su extensa disponibilidad de elementos de modelación. La especificación de UML permite a los desarrolladores adaptarlo a través de extensiones para satisfacer requerimientos particulares.

Cuando UML fue creado, sus autores consideraron la semántica para satisfacer el dominio de las aplicaciones que hasta ese entonces existían. Sin embargo esta semántica ha sido insuficiente conforme los conceptos tecnológicos evolucionan, tal es el caso de Web Services. Para solucionarlo proponemos mecanismos de extensión que permitan a los desarrolladores ampliar la semántica de UML, definiendo nuevos elementos aplicables a Web Services mediante estereotipos. UML no define cómo desarrollar software, es por eso que se requiere de un proceso que identifique las etapas de desarrollo, las actividades y los artefactos. En este documento presentamos la propuesta de un proceso con estas características y mostramos su validación mediante dos diferentes casos de estudio.

1.5. ¿Qué significa Web Services?

Los días 16 y 17 de enero del 2002 [Cla2002] se llevó a cabo en la ciudad de San Francisco, la conferencia *"Next-Generation Web Services"* reuniendo a desarrolladores, investigadores y empresarios, donde se confirmó la inmadurez existente en la industria de la Tecnología Informática con respecto al desarrollo de Web Services. Un tópico discutido fue la definición del término Web Services, generalizando tres acepciones:

1. Web Services es la siguiente generación de aplicaciones Internet basadas en servicios.
2. Web Services representa arquitecturas de software basadas en componentes.
3. Web Services proporciona un mecanismo para que las aplicaciones se comuniquen entre sí a través de Internet.

Esa diversidad de acepciones nos permite ver que el concepto Web Services aún no esta definido completamente y más allá de eso, ninguna de las tres nos define el concepto claramente. Las dos primeras resultan ambiguas, redundantes en el concepto y poco descriptivas, la tercera nos da una idea más clara, sin embargo no se define exactamente que se entiende por "mecanismo". A continuación se presenta una definición que presenta una idea clara de lo que significa Web Services en relación a la integración de aplicaciones.

En [LF2002] se define el término *Web Services*, como una interface que describe una colección de operaciones que están accesibles en la red a través de estándares abiertos, descrito por medio de una *descripción del servicio.* La descripción del servicio [Kre2001] contiene los detalles de la interface para interactuar con el servicio. Es decir, los tipos de datos, las operaciones, la información de enlace y la ubicación de la red. Lo anterior podría también incluir categorizaciones y otros metadatos para facilitar el descubrimiento y la utilización del servicio. Un *servicio* [Kre2001] es la implementación del Web Services, es decir la plataforma de desarrollo, las herramientas, el marco de trabajo, entre otros; un servicio es un módulo de software el cual esta disponible por el proveedor del servicio. La definición anterior es en la que se fundamenta el trabajo presentado en este documento.

La conceptualización de Web Services es inherente a los estándares abiertos. Los Web Services utilizan registros que publican servicios que pueden ser descubiertos e invocados y que mantienen directorios de servicios que pueden ser agregados, modificados o eliminados dinámicamente (UDDI), un estándar que describe los servicios e involucra interfaces Web Services (WSDL) y un protocolo de comunicación de objetos distribuidos que provee de un mecanismo no restringido de envío de paquetes de mensajes de los procesos (SOAP); estos estándares son detallados en sección 2.1.6 posteriores.

1.6. Organización de la tesis

Este documento es el resultado del trabajo realizado sobre una propuesta de procesos para desarrollar Web Services y su estructura es la siguiente:

Capítulo 1: presenta las generalidades acerca de este documento, destacando la especificación del objetivo de este trabajo y se define la problemática identificada y la motivación. Se presenta además una definición del concepto Web Services.

Capítulo 2: se explican tres tópicos sobre los que se fundamente este trabajo. Web Services, donde se explica porque es relevante, cuales son sus características, la arquitectura y las tecnologías involucradas. UML, se explica el significado de este concepto y se especifican los mecanismos de extensión. Estado del arte: metodologías propuestas, en esta sección se presentan las propuestas sobre las cuales se fundamenta el proceso propuesto en esta tesis, se explican las metodologías presentadas por cada autor.

Capítulo 3: se presenta el proceso propuesto. Se identifican las actividades, los artefactos que se deben modelar en cada fase del proceso propuesto.

Capítulo 4: se valida la propuesta presentada mediante dos casos de estudio. En este capítulo se presentan los modelados de cada caso de estudio, siguiendo el proceso presentado.

Capítulo 5: se presentan algunos trabajos relacionados a este trabajo de tesis para identificar que es lo que otras personas están haciendo respecto a esta área, con la finalidad de identificar el estado del arte actual.

Capítulo 6: finalmente se muestran los resultados y las conclusiones a los que se llegaron mediante este trabajo realizado.

Adicionalmente se presentan las referencias y dos anexos. En el anexo A, se presenta una breve información acerca de cada autor de las metodologías presentadas en el capítulo 2 y en el anexo B, se presentan los patrones de diseño (GRASP) presentados por Larman.

Capítulo 2 Marco Conceptual

Este capítulo define los fundamentos de este trabajo y se encuentra dividido en tres partes. En la primera se presenta un panorama general de los aspectos fundamentales que existen alrededor de Web Services, tales como definir el concepto e identificar los requerimientos, arquitectura y tecnologías, proporcionando una perspectiva de la relevancia de Web Services en la industria. En la segunda parte se define UML., explicando qué es UML para identificar qué se puede hacer mediante este lenguaje, por qué es ampliamente utilizado y se explican los mecanismos de extensión de Unified Modeling Languaje (UML) y en la tercera se presenta un compendio de las metodologías propuestas por cuatro diferentes autores, las cuales fueron analizadas para identificar si era posible desarrollar Web Services y sobre las cuales se fundamenta el proceso propuesto.

2.1 Web Services

Esta sección presenta siete subsecciones, mediante las cuales se pretende identificar cada aspecto relevante de Web Services, tales como características, arquitectura, requerimientos, tecnologías y su importancia dentro de la industria.

2.1.1. ¿Cuál es la relevancia de los Web Services?

La Web ha sido considerada como una forma de compartir y distribuir información de manera global [GSBDDNN2002]. La interacción de aplicaciones con aplicaciones basadas sobre ella ha empezado a tomar mayor importancia, primero en relación con el desarrollo de negocios sobre el Web y automatización de transacciones de negocios con negocios y en la actualidad con la finalidad de compartir recursos.

Los Web Services representan un avance en la industria de las Tecnologías de la Información debido a que integran aplicaciones de dispositivos como computadoras personales, servidores Web, bases de datos y redes virtuales [Vau2002].

2.1.2. Ejemplos de Web Services

Los Web Services toman gran relevancia dentro de las operaciones de una organización, ya que en el entorno que brindan, es posible descubrir otros componentes y utilizarlos para sus transacciones de negocios [RR2001].

Especialmente Web Services puede representar la diferencia en aplicaciones como:

- Servicios de tarjeta de crédito, dónde las transacciones del proceso de crédito deben proporcionar un número de cuenta específico.
- En un servicio de aerolíneas dónde es necesario tener un registro de los vuelos, disponibilidad y reservación.
- Búsquedas de información.
- Fusión de información ubicada en diferentes sitios Web.
- Mercados electrónicos (por ejemplo subastas).
- Consultas usando lenguaje natural.

Estos son sólo algunos de los ejemplos dónde podemos ver que el desarrollo de aplicaciones Web Services ya está siendo utilizado.

2.1.3. Características principales de Web Services

Los Web Services se distinguen por tres características principales: Independencia de plataforma, utilización de estándares abiertos e interoperabilidad.

Independencia de plataforma: los Web Services ocultan los detalles de implementación del servicio, es decir, es independiente de plataformas de hardware, software y del lenguaje de programación. Esto permite que las aplicaciones que utilizan Web Services sean débilmente acopladas, orientadas a componentes e independiente de plataformas.

Utilización de estándares abiertos: como SOAP, WSDL, UDDI (detallados en la sección 2.1.6) que al estar disponibles públicamente facilitan que los Web Services puedan comunicarse entre sí.

Interoperabilidad: La independencia de plataforma y el uso de estándares abiertos, dan lugar a esta característica que es la que permite la comunicación entre ellos.

2.1.4. Requerimientos de Web Services

Básicamente los requerimientos de Web Services pueden ser divididos en tres grupos: de integración, tecnológicos y orientados al servicio.

Requerimientos de Integración

Se refiere al conjunto de entidades y recursos que identifican a cada servicio.

Una entidad son las aplicaciones que participarán en la integración y los recursos es aquello que cada aplicación compartirá.

Requerimientos Tecnológicos

Los requerimientos tecnológicos son las especificaciones que se deben considerar para construir Web Services, a nivel tecnología. Estos consisten en:

Protocolo: Mecanismo de comunicación utilizado para enviar mensajes.

Registro y descubrimiento de servicios: Estándar abierto de publicación de registros para que puedan ser descubiertos e invocados y que mantendrán directorios servicios que podrán ser agregados, modificados o eliminados dinámicamente

Descripción del servicio: Estándar que especifica la descripción de las interfaces Web Services

Requerimientos orientados al servicio

Estos requerimientos son el conjunto de atributos relacionados directamente con el servicio para lograr que los servicios estén disponibles. Los atributos principales son:

1. Seguridad: Web Services realiza intercambio de mensajes a través de la red, durante el compartimiento de información. Esto conlleva algunos riesgos de seguridad, por lo que deben ser satisfechos los siguientes requerimientos, para que el intercambio de información entre Web Services sea realizado en forma segura [NHN2002]:

    a) Confidencialidad: para asegurar que el contenido de los mensajes no esté expuesto a intrusos.

    b) Autenticación: para garantizar que el acceso a las aplicaciones y a los datos esté restringido solamente para quienes se identifiquen adecuadamente.

    c) Integridad de los datos: para asegurar que los mensajes no sean modificados accidentalmente o deliberadamente durante el intercambio.

    d) Prueba de origen: para asegurar la identidad del origen de un mensaje o datos.

    e) Autorización: para asegurar que el remitente esté autorizado para enviar el mensaje.

2. Confiabilidad: Al realizar solicitudes de servicios entre Web Services, se esperará que el servicio esté disponible, que las solicitudes sean enviadas una sola vez y que en caso de existir fallas en el envío de la solicitud, existan mecanismos de información [LF2002]. También se requiere de especificar información para almacenar las solicitudes de forma correcta, saber cuando almacenarlas y cómo enviar solicitudes de forma adecuada.

2.1.5. Arquitectura orientada al servicio

Una arquitectura Web Services típica consiste de tres entidades [RR2001], tal como se muestra en la figura 2.1:

1. Proveedores de servicios quienes crearán Web Services y los publicarán mediante registros de los servicios.
2. Repositorios de servicios quienes mantendrán registros de los servicios publicados.
3. Solicitadores de servicios son los que invocan al servicio.

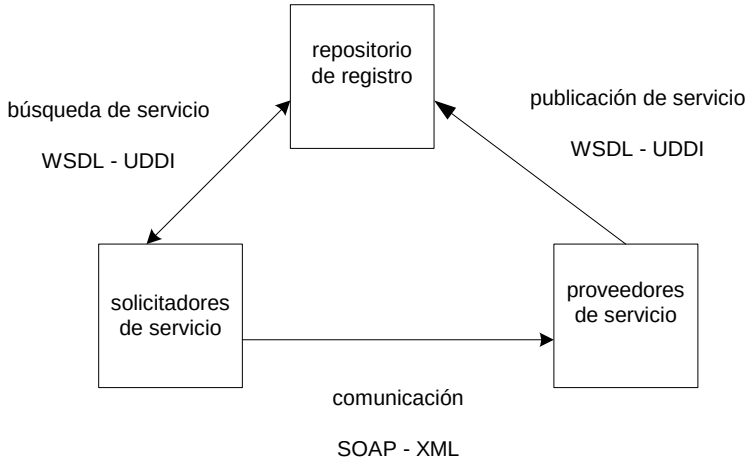


Figura 2.1. Arquitectura orientada al servicio

Web Services da soporte a arquitecturas orientadas al servicio [Vin2002] esto es, un servicio con una interface definida y métodos que definen el intercambio de datos para guardar registros en directorios distribuidos de servicios donde las aplicaciones pueden buscar detalles de la interacción entre servicios.

En la interacción de la arquitectura antes definida se identifican tres operaciones básicas como se muestra en la figura 2.1, las cuales son necesarias para publicar la descripción del servicio a través de WSDL, buscar y encontrar la descripción del servicio mediante UDDI o intercomunicar al proveedor y al solicitador de los servicios usando SOAP. Estas operaciones son definidas a continuación:

1. Publicación de servicio: para que el servicio de descripción sea accesado requiere ser publicado en el repositorio de registro para que el solicitador del servicio pueda encontrarlo.

2. Búsqueda de servicio: en esta operación el solicitador del servicio obtiene la descripción del servicio directamente o mediante consultas al repositorio del registro. Esta operación puede ser utilizada de dos maneras diferentes: definida en tiempo de diseño (servicios privados) o bien en tiempo de ejecución (servicios públicos).

   Obtener la descripción del servicio dependerá de la publicación del servicio y del dinamismo del Web Service. En otras palabras, la búsqueda del servicio se puede realizar en dos diferentes fases del desarrollo: en tiempo de diseño y en tiempo de ejecución. En tiempo de diseño el solicitador del servicio busca la descripción del servicio en las interfaces conocidas (Web Services con los que se está integrando). En tiempo de ejecución, el solicitador del servicio busca un Web Service basado en la comunicación o en los servicios publicados.

   Con el enfoque de publicación de interfaces conocidas, el solicitador de servicio obtiene la descripción del servicio en tiempo de diseño para utilizarlo en tiempo de ejecución. En este caso, la descripción del servicio puede ser representada en la lógica del programa, almacenándola en un archivo o un repositorio local de descripciones de servicio [Kre2001]. Esta es la forma más común de implementar las búsquedas del servicio; es decir, dejar que un Web Service acceda a servicios públicos no esta siendo explotado actualmente, se requiere de mayor programación y no se tiene la certeza de que ese servicio pudiera ser encontrado. Contrario a utilizar servicios privados donde ya se tiene la certeza de que el servicio estará disponible.

3. Comunicación: un servicio requiere ser invocado. En esta operación el solicitador invoca o inicia la interacción con el proveedor, usando la descripción del servicio para conocer los detalles de ubicación, contacto e invocación.

2.1.5.1. Representación de la arquitectura

La arquitectura de Web Services debe ser independiente del número de servicios involucrados y de la plataforma de desarrollo [Oell2001], puede representarse mediante dos vistas: vista de pila y vista lógica, las cuales se describen a continuación.

Vista de pila

Los vendedores principales de Web Services (ver sección 2.1.7) definen la arquitectura de Web Services en una pila de capas [Mye2001], es por eso que esta pila puede variar de un vendedor a otro, sin embargo en esencia deben definirse las siguientes capas tal como se muestra en la figura 2.2.

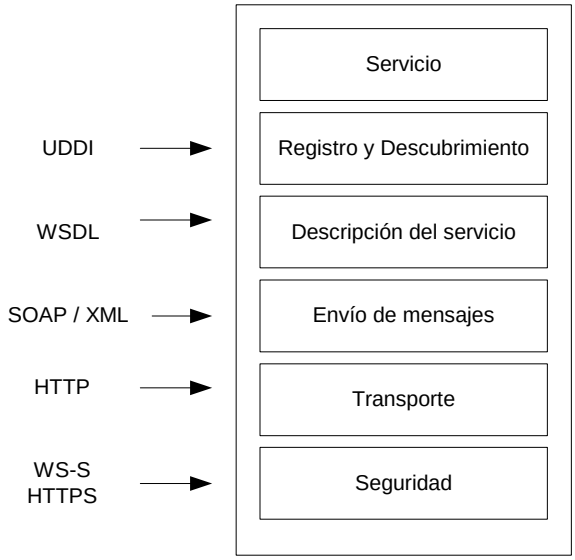


Figura 2.2 Pila de capas

Capas

Tanto el solicitador de servicio como el proveedor de servicio tienen implementada esta pila de capas:

1. Seguridad: Se implementan mecanismos de seguridad como integridad, confidencialidad y autenticación de los mensajes. Estos mecanismos pueden ser implementados utilizando tecnologías como Web Services - Security (WS-S) o Secure Hypertext Transfer Protocol (HTTPS).
2. Transporte: En esta capa se implementan los protocolos como Hypertext Transfer Protocol (HTTPS) que permitirán que los Web Services puedan invocar o proveer los servicios.
3. Envío de mensajes: Esta capa es implementada sobre la capa de transporte para que se puedan realizar las operaciones de búsqueda de servicios, publicación de servicios y comunicación (ver sección 2.1.5), para lo cual se implementa SOAP con mensajes XML.

4. Descripción del servicio: Esta capa proporciona los mecanismos de invocación de los servicios. WSDL es el estándar para describir en formato XML la implementación y las interfaces del servicio.
5. Registro y descubrimiento: Esta capa permite que los proveedores puedan publicar sus servicios mediante registros UDDI o bien que los solicitadores puedan invocarlos y encontrarlos. Estos registros básicamente describen la implementación del servicio.
6. Servicio: En esta capa se definen los servicios que se compartirán.

Vista lógica

En la vista lógica para Web Services se identifican las invocaciones necesarias a aplicaciones para satisfacer las solicitudes del servicio.

En [Oell2001] Oellerman identifica diferentes formas de satisfacer un servicio, en otras palabras, un servicio puede ser compuesto de las siguientes maneras:

1. Un Web Service puede requerir enlazarse con otro Web Service y este a su vez requerir enlazarse con otro y así hasta satisfacer una petición a un servicio.
2. Un Web Service puede ser satisfecho por la unión de varios Web Services. Esta unión es considerada como una misma aplicación.
3. Un Web Service es satisfecho por varios Web Services, donde el servicio se compone por la combinación de los casos anteriores, es decir pueden ser servicios en cadena o bien servicios que funcionen como una sola aplicación.

2.1.6. Tecnologías

El núcleo de las tecnologías Web Services como se describe en [RR2001] lo forman: Web Services Description Language (WSDL); Universal Description, Discovery and Integration (UDDI); y Simple Object Access Protocol (SOAP).

Web Services habilita la interoperabilidad a través de estándares abiertos [Vau2002], XML es el estándar más importante para aplicaciones Web Services y es la base para el resto de las tecnologías Web Services. XML es un metalenguaje que permite a los usuarios definir sus propias etiquetas, que proporcionan

información sobre las plataformas de desarrollo. Lo cual permite la comunicación independientemente de la plataforma e integrar diferentes tipos de datos en los sistemas de las organizaciones.

Las aplicaciones deben utilizar SOAP [Vau2002] para invocar servicios de un Web Service. SOAP permite que aplicaciones que se ejecutan sobre diferentes sistemas operativos se comuniquen entre sí, a través de mecanismos que utilizan HTTP y XML. Es esta la razón por la que un Web Service sólo requerirá compatibilidad con SOAP para trabajar con otros Web Services.

WSDL permite que los Web Services especifiquen su descripción del servicio y determina los detalles específicos del protocolo de comunicación [Vau2002]. WSDL se utiliza para describir una lista de los Web Services registrados a través de UDDI, para que las aplicaciones puedan accesar a ellos.

2.1.7. Principales vendedores de Web Services

En [Vau2002] se muestran algunas tecnologías relacionadas con Web Services, donde se identifica que los principales vendedores son BEA Systems, Hewlett-Packard, IBM, Oracle, Sun Microsystems y Microsoft quienes están trabajando sobre servicios Web, desarrollando infraestructuras Web Services para un mejor rendimiento y utilización de sus propios productos o de sus socios. Las características de las tecnologías de estas compañías son:

- .NET es un conjunto de tecnologías de Microsoft para conectar información, personas, sistemas y dispositivos. Facilita la integración a través del uso de Web Services.
- La plataforma Web Services de HP comenzó con el proyecto e-Speak, el cual originalmente fue diseñado para desarrollar Web Services usando tecnologías creadas por HP. Sin embargo, el crecimiento de XML para desarrollar Web Services disminuyó su importancia. Actualmente HP se enfoca a desarrollar Web Services con especificaciones XML y recientemente con J2EE.
- IBM está incluyendo aplicaciones con Web-Sphere, un kit de desarrollo de software que define para arquitecturas de anteproyecto, herramientas, componentes, demos y ejemplos de ayuda que diseñan y ejecutan aplicaciones Web Services.
- Oracle ha desarrollado Oracle9iAS Web Services, es cual forma parte de la suite Oracle9i/Developer. Proporciona un ambiente de desarrollo Web

Services utilizando J2EE. Mediante esta herramienta es posible publicar, crear y consumir Web Services basado en SOAP, WSDL y UDDI.

- BEA creó BEA WebLogic Enterprise Platform, mediante el cual es posible crear Web Services. WebLogic permite integrar aplicaciones y facilita la integración de procesos de negocios entre ellas.
- SUN ha desarrollado Sun ONE Web Services Platform Developer Edition. Una suite de herramientas para la integración de aplicaciones desarrolladas utilizando Java y esta basado en SOAP, WSDL y UDDI.

2.2 UML

Esta sección tiene dos subsecciones en la primera se identifica que es UML, que no es UML y dónde puede ser utilizado y en la segunda se explican los mecanismos de extensión existentes en la especificación de UML.

2.2.1. ¿Qué es UML?

Unified Modeling Language (UML) es un lenguaje para visualización, especificación, construcción y documentación de artefactos de un sistema de software intensivo [OMG2001]. UML permite estandarizar el modelado de procesos de negocios, funciones del sistema, esquemas de bases de datos y componentes reusables de software.

Conallen, quien ha utilizado UML para modelar aplicaciones Web [Mc2001], afirma que “un sistema de complejidad no trivial requiere ser modelado, para ser entendido correctamente y realizado en la etapa de diseño”[Con1999]. La decisión de utilizar un lenguaje de modelado está basada en los requerimientos de cada aplicación [Con2000]. Con la aceptación de UML por parte de OMG (lenguaje oficial de modelado de objetos), cada vez más desarrolladores de aplicaciones están utilizando notación UML en el modelado de sistemas.

UML es solamente un lenguaje, por si sólo no representa una metodología o proceso, es decir, se requiere de una guía de desarrollo. UML proporciona un vocabulario y reglas para la representación física y conceptual de un sistema [BRJ1999], pero no define qué modelos deben ser creados y cuándo deben ser realizados. Estas especificaciones las realiza el proceso de desarrollo [BRJ1999], un proceso deberá guiar al desarrollador en la identificación de qué artefactos

producir, cuales actividades realizar y cuándo realizarlos, para que el resultado final de todas estas especificaciones coincidan en el desarrollo del modelado de un sistema.

Como ya se mencionó UML no es un proceso por sí mismo, sin embargo puede ser utilizado en procesos basados en casos de uso, centrados en la arquitectura o en procesos iterativos e incrementales [BRJ1999]. En otras palabras:

- Basado en casos de uso significa que los casos de uso son utilizados como principal artefacto para definir el funcionamiento del sistema, para verificar y validar la arquitectura del sistema.
- Centrado en la arquitectura significa que la arquitectura del sistema es usado como principal artefacto para la conceptualización, construcción y administración del sistema.
- Un proceso iterativo significa que el desarrollo se realiza en varios ciclos, un proceso incremental significa que se agregan funcionalidades en cada iteración.

2.2.2. Mecanismos de extensión

UML es un lenguaje estándar para modelar sistemas, sin embargo no es posible que desde su creación haya considerado todos los elementos y expresiones necesarias para modelar los diferentes tipos de sistemas a través del tiempo. Es decir, cuando fue creado se consideró modelar los sistemas que se conocían hasta ese momento, pero obviamente no cubría las características de los sistemas que han surgido últimamente, es por eso que se establecieron mecanismos de extensión.

Estos mecanismos de extensión son: estereotipos, etiquetas y condiciones. Un estereotipo extiende el vocabulario de UML, permite crear nuevos tipos de elementos básicos pero aplicados al problema en particular. Las etiquetas extienden las propiedades de los elementos de UML, permitiendo crear nueva información a la especificación de cada elemento. Y las condiciones extienden la semántica de UML, permiten agregar nuevas reglas o modificar las existentes. En [BRJ1999] se puede encontrar ampliamente esta información de manera detallada.

2.3 Estado del arte: metodologías propuestas

Esta sección tiene dos subsecciones en la primera se identifican las similitudes y diferencias entres las metodologías propuestas por Jim Conallen, Doug Rosenberg y Kendall Scott, Craig Larman y, Bernd Oestereich (ver anexo A) y en la segunda se detallan las cuatro propuestas de desarrollo de software analizadas.

2.3.1 Similitudes y Diferencias Generales

En esta sección se identifican las principales semejanzas y diferencias en relación al enfoque en que cada autor se basa para proponer su metodología:

- Conallen se basa en la utilización de dos procesos: Rational Unified Process (RUP) e ICONIX.
- Oestereich propone un proceso de desarrollo de aplicaciones orientado a objetos usando Unified Modeling Language.
- Larman presenta una propuesta de desarrollo de aplicaciones utilizando Unified Modeling Language.
- Rosenberg/Scott utilizan ICONIX en su propuesta de desarrollo.

UML es la herramienta utilizada por las cuatro propuestas para representar gráficamente cada etapa del proceso. La diferencia principal en las cuatro propuestas es el proceso en el cual se basan: RUP o ICONIX.

RUP es iterativo e incremental, se centra en la arquitectura y es dirigido por los casos de uso definidos, mientras que el proceso ICONIX se enfoca principalmente en la realización de un análisis formal de robustez y utiliza un gran número de casos de uso simples [Con2000].

Las dos propuestas más contrastantes son las presentadas por Conallen y por Rosenberg/Scott. Conallen se basa en ambos procesos con lo cual enriquece el desarrollo de las aplicaciones, sin embargo para Rosenberg/Scott RUP tiene un uso empírico y su propuesta la basa principalmente en ICONIX, pues él es el creador de dicho proceso. Por otro lado Oestereich no especifica la utilización de un proceso en particular, mientras que Larman en la primera edición de [Lar1999] no especifica la utilización de un enfoque en particular pero presenta un proceso iterativo; en la segunda edición se basa en Rational Unified Process (RUP).

Las cuatro metodologías analizadas utilizan un proceso iterativo e incremental. Lo cual significa que cada fase es repetida y refinada hasta cubrir por completo los requerimientos del sistema. No obstante el enfoque de cada propuesta es la diferencia en los cuatro procesos. En las siguientes secciones se detallan cada una de las cuatro metodologías.

2.3.2 Descripción de las metodologías analizadas

2.3.2.1 Larman

Larman propone en [Lar1999] un proceso iterativo; para demostrar su propuesta utiliza una aplicación de ejemplo, sobre la cual efectúa dos iteraciones: la primera está enfocada en una simple aplicación de las funciones básicas y la segunda se destina a ampliar la funcionalidad del sistema.

Larman divide el proceso de desarrollo que propone en las siguientes fases: análisis, diseño y construcción.

A continuación se detallan las fases para cada iteración en el proceso propuesto por Larman:

1ª. Iteración

- Requerimientos

Larman propone identificar los casos de uso y utilizar diagramas de casos de uso, como se muestra en la figura 2.3 donde se observa la identificación de un actor relacionado con tres casos de uso. Sugiere también dos actividades: realizar una descripción textual para cada caso de uso y la clasificación de los casos de uso en base a establecer una jerarquización en los procesos definiendo como primarios los que son comunes, secundarios los que son raros y opcionales.

Se definen las especificaciones de los requerimientos y de los casos de uso. Se sugiere la creación de un modelo conceptual y de un prototipo de arquitectura preliminar, sin embargo el objetivo principal es establecer una agenda de construcción e implementación de los casos de uso. Al concluir esta etapa se da inicio a la realización del análisis y diseño orientado a objetos.

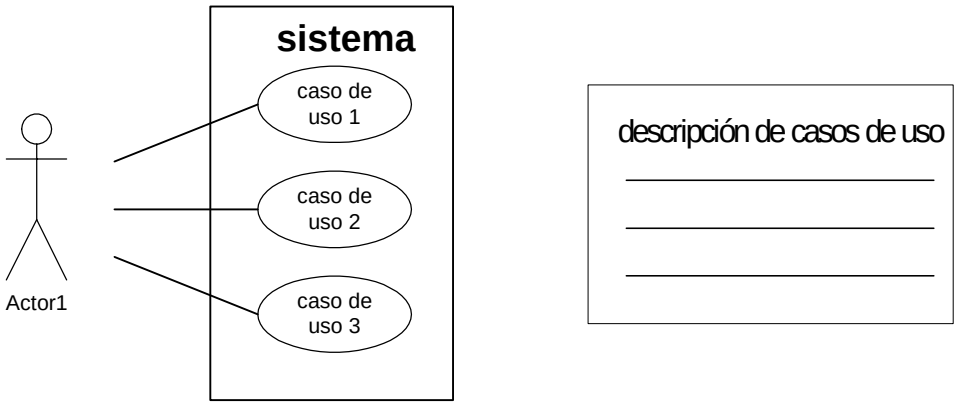


Figura 2.3. Modelación de requerimientos (Larman)

ï Análisis

Inicia con la construcción de un modelo conceptual, el objetivo es identificar los conceptos significativos en el dominio; es el artefacto más importante para esta fase. El modelo conceptual identifica los conceptos (Rumbaugh et al [Vin2002], les llama clases), las asociaciones entre ellos y sus atributos. En la figura 2.4 se identifican cuatro conceptos con sus respectivos atributos, se observa además que el concepto1 se relaciona al concepto2 por medio de la asociación iniciado-por ; el concepto1 también se relaciona con el concepto3 y el concepto4 por medio de almacena y contenida en, respectivamente.

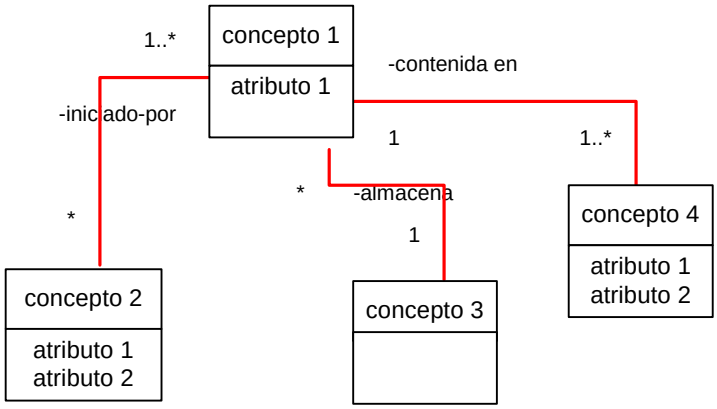


Figura 2.4. Modelo conceptual en la fase del Análisis (Larman)

Definir un glosario de los términos del dominio es relevante para estandarizar el lenguaje técnico. Este glosario debe iniciarse en la etapa de análisis, pero deberá ser modificado a lo largo del proceso de desarrollo conforme se identifiquen nuevos términos.

Después se define el comportamiento del sistema mediante diagramas de secuencia, para describir qué es lo que hace el sistema. Para la construcción de

estos diagramas se utilizan los casos de uso ya descritos y se identifican los eventos que originan cada caso de uso. Un evento es un hecho externo de entrada que un actor produce en un sistema y da como origen a una operación de respuesta. Algunos eventos requerirán parámetros tal como se muestra en la figura 2.5 en el evento1 y evento3. El objetivo de estos diagramas es especificar los eventos a los que responderá el sistema y las responsabilidades y poscondiciones que tiene sus respectivas operaciones.

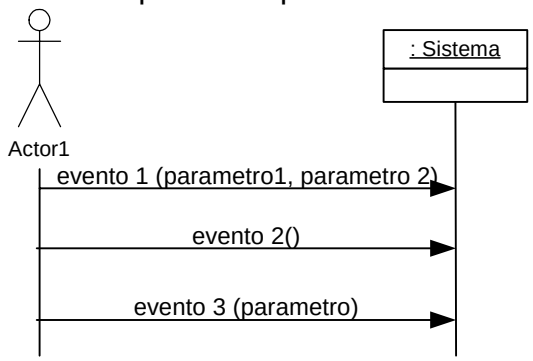


Figura 2.5. Modelación de diagramas de secuencia en la fase del Análisis (Larman)

Finalmente en esta primera iteración de la fase del análisis, Larman establece la necesidad de crear contratos para las operaciones de un sistema. Básicamente un contrato es la descripción detallada de cada operación, identificando el conjunto de condiciones necesarias que se deben satisfacer para la correcta ejecución de cada operación, al igual que las condiciones que deben ser satisfechas por la operación. A estos conjuntos de condiciones se les define como pre- y pos-condiciones.

- Diseño

La transición a la fase del diseño inicia con la descripción de casos reales de uso, es decir, definir un diseño concreto de cómo se realizará el caso. En otras palabras, describir casos de uso en los cuales existirá intervención directa del usuario.

La siguiente actividad propuesta en [Lar1999] para el diseño es la descripción de diagramas de colaboración, es decir, representar la interacción entre objetos usando envío de mensajes.

Se pueden utilizar diagramas de colaboración para describir las interacciones entre los objetos en un formato de grafo o red, como se muestra en la figura 2.6, donde :claseAIntencia recibe el mensaje1 de entrada y le envía dos mensajes (mensaje2 y mensaje3) a :ClaseBInstancia. En [Lar1999] se prefieren estos diagramas ya que resultan más adecuados para expresar información contextual y la lógica condicional y la concurrencia. Para Larman los diagramas de interacción

constituyen uno de los artefactos más importantes que se generan en el análisis y en el diseño orientado a objetos.

Larman propone la utilización de patrones GRASP (General Responsability Asignment Softwre Patterns), de los que es el creador, para asignar responsabilidades. GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones [Lar1999]. Los principales patrones GRASP sugeridos por Larman se describen en el anexo B.

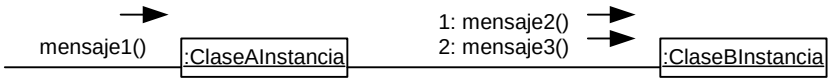


Figura 2.6. Modelación de diagramas de colaboración en la fase del Diseño (Larman)

Después se realiza un refinamiento de los diagramas de clases, con el objetivo de identificar las clases, los métodos y las asociaciones. La diferencia con los diagramas conceptuales especificados en la fase del análisis es que en esta fase se definen los métodos que cada clase deberá contener.

También en el diseño se elabora la arquitectura de la aplicación. La arquitectura dependerá de la aplicación, sin embargo, los elementos básicos de una arquitectura por capas, son: presentación, lógica de aplicaciones y almacenamiento. La arquitectura es representada mediante diagramas de paquetes, donde cada capa es modelada con paquetes, tal como se muestra en la figura 2.7, donde cada capa es representada por paquetes; se identifican además la modelación de las relaciones entre las capas.

Los paquetes son utilizados para agrupar subsistemas pertenecientes a cada capa. En esta primera iteración del diseño, la arquitectura resulta ser general es decir, no tiene demasiados detalles. Cada paquete agrupa los elementos necesarios para brindar un conjunto de servicios comunes, con un nivel relativamente alto de acoplamiento y colaboración.

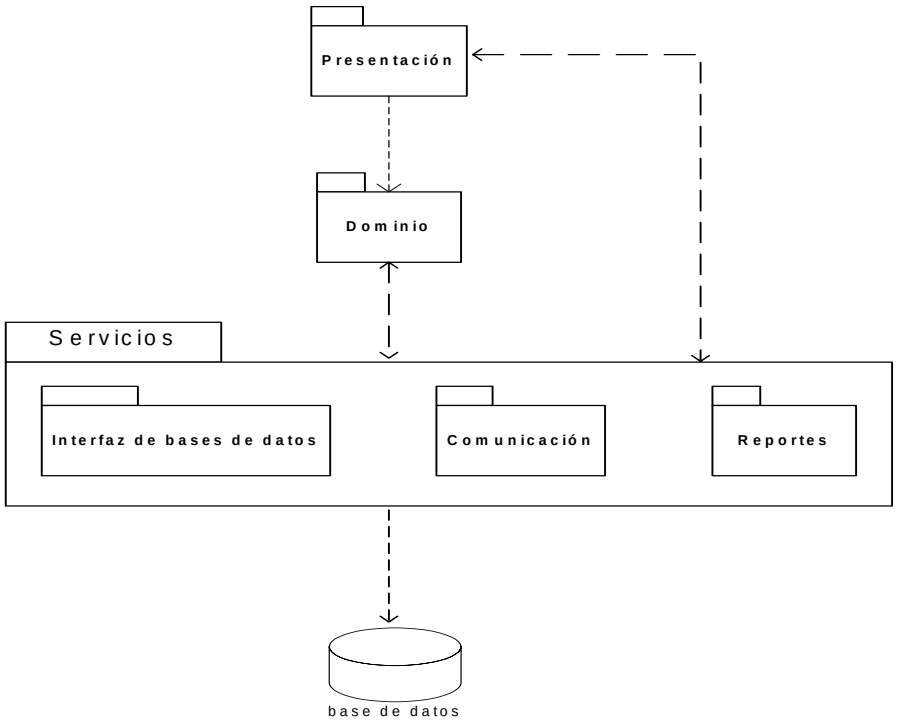


Figura 2.7. Modelación de la arquitectura en la fase del Diseño (Larman)

- Construcción

Para escribir el código se debe basar en los diagramas de colaboración y los diagramas de clases definidos. Es claro que los detalles del código dependerán completamente del lenguaje utilizado.

2ª. Iteración

- Análisis

La segunda iteración utiliza los casos de uso definidos en la primera iteración, pero se les agrega una mayor funcionalidad para que se aproximen al caso de uso final y completo. Se utiliza el mismo caso de uso en las diferentes iteraciones, pero depurándolo y bajo diferentes enfoques. Larman propone, en esta segunda iteración, relacionar los casos de uso, es decir, conectar los diferentes casos de uso mediante asociaciones.

La siguiente actividad sugerida por Larman es identificar nuevos conceptos a partir de la lista de categorías de conceptos para incluirlos en la segunda iteración. Larman propone refinar el modelo conceptual, añadiendo atributos a las asociaciones.

Se definen diagramas de secuencia del sistema y los contratos de la operación. Al igual que en la primera iteración, el objetivo de estos diagramas es representar a manera de secuencia las especificaciones de los casos de uso correspondientes.

Como actividad final del análisis, se propone elaborar diagramas de estado para los casos de uso, es decir, un diagrama de estado para casos de uso describe los eventos globales del sistema y su secuencia en un caso de uso. Consiste en describir la secuencia permitida de los eventos externos del sistema que son reconocidos en el sistema dentro del contexto de un caso de uso.

- Diseño

Se siguen utilizando los patrones GRASP, con la finalidad de representar de la forma más clara posible la aplicación, de modo que la transición a la codificación sea automática prácticamente.

- Construcción

Al igual que en la primera iteración, para la realización de esta fase los diagramas definidos durante el diseño deben ser utilizados para efectuar la escritura del código. Es decir, los diagramas deben ser lo suficientemente claros y precisos de manera que faciliten la codificación del sistema.

2.3.2.2 Oestereich

El proceso de desarrollo presentado por Oestereich en [Oes1999] se divide en: Análisis de requerimientos, análisis del Dominio del problema y desarrollo iterativo e incremental, donde cada fase tiene sus propias actividades. Estas fases difieren completamente de los nombres de las fases de las propuestas de los otros autores. Una comparación que se puede realizar es que el análisis de requerimientos corresponde con la fase de requerimientos por otros autores, el análisis del dominio se refiere a las fases de análisis y diseño identificadas por el resto de los autores. Por otro lado la fase de desarrollo iterativo e incremental es propiamente la implementación y administración del proceso de desarrollo. A continuación se describen las fases propuestas en [Oes1999]:

- Análisis de requerimientos:

a) Estudio preliminar. Mediante el cual se debe describir la naturaleza de cada tarea o del problema y, se debe considerar una primera solución y alternativas.

b) RAD (Rapid Analisis and Design). Los más importantes procesos de negocio, casos de uso, objetos del negocio y componentes son contemplados. Se elaboran prototipos del diseño y de la implementación de la aplicación.
c) Análisis de los procesos de los negocios. Se efectúa con la finalidad de analizar el contexto en el que será implementada la aplicación.
d) Análisis de los casos de uso. Identificará lo qué el sistema debe hacer. En la figura 2.8 se muestra este diagrama un diagrama de casos de uso y su respectiva descripción.

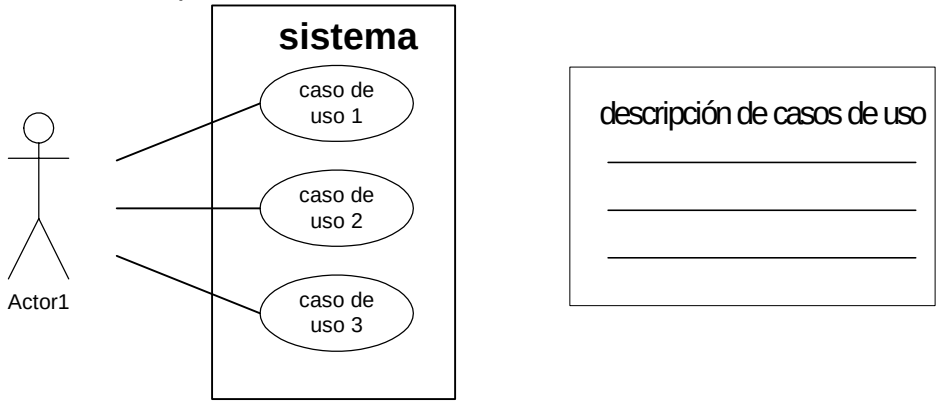


Figura 2.8. Modelación de casos de uso en la fase del Análisis de requerimientos (Oestereich)

e) Evaluación de la arquitectura y del ambiente de desarrollo de software. Reducirá los riesgos tecnológicos e identifica las deficiencias del desarrollo de la aplicación. Oestereich sugiere representar la arquitectura agrupando en componentes y estableciendo relaciones de dependencia, donde cada componente podría representar un módulo del sistema. En la figura 2.9 se muestra un componente que implementa dos interfaces a través de las cuales se relaciona con los otros componentes.

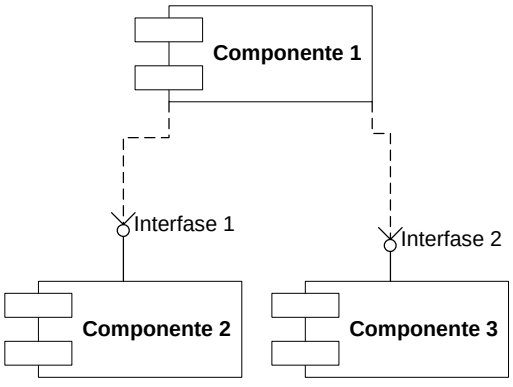


Figura 2.9. Modelación de la arquitectura en la fase del Análisis de requerimientos (Oestereich)

ï Análisis del dominio del problema:

a) Diagramas de actividad: Oestereich propone utilizar la definición de los casos de uso y crear el diagrama de actividad correspondiente. Los diagramas de actividad proporcionarán una representación más detallada como procesos paralelos, dependencias, puntos de decisión, etc. En la realización de un caso de uso generalmente se requieren de actividades por parte de diferentes actores, cada actor tendrá una responsabilidad específica. En la figura 2.10 se muestra un diagrama de actividad dividido en dos partes, lo que significa que dos actores están involucrados, cada parte representa la responsabilidad que tienen los actores, para la realización del caso de uso que está representando el diagrama.

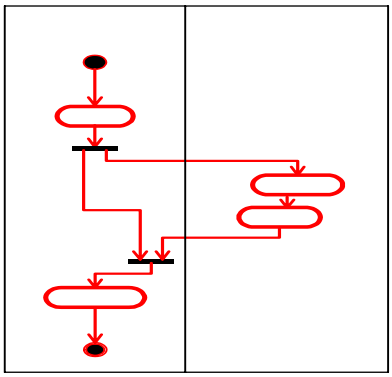


Figura 2.10. Diagrama de actividad utilizado en la fase del Análisis del dominio (Oestereich)

b) Para modelar las clases del negocio resultan útiles las tarjetas CRC y modelos de referencia, se propone la utilización de diagramas ER. Lo importante en este modelado es identificar las clases (objetos del negocio) y sus relaciones, aún cuando aún no se sugiere un modelado formal, es relevante la identificación de la estructura del dominio.

c) El modelado de las clases del dominio es representado por diagramas de clase con el propósito de identificar los atributos, operaciones, condiciones y relaciones de las clases del dominio, tal como se muestra en la figura 2.11.

d) En la definición de los componentes y los subsistemas se propone la utilización de modelos de paquetes para identificarlos, considerando aspectos técnicos para minimizar la dependencia e interfaces innecesarias entre los componentes individuales.

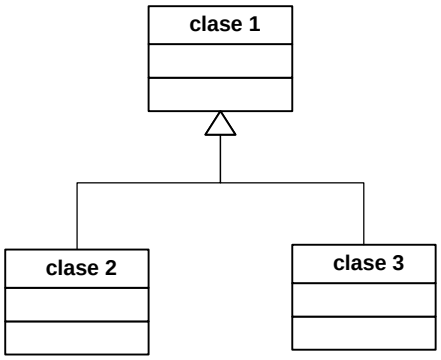


Figura 2.11. Modelación de las clases del dominio en la fase de Análisis de requerimientos (Oestereich)

- Desarrollo iterativo e incremental:

Las actividades propuestas por Oestereich en esta etapa, básicamente pueden ser resumidas en un concepto de administración, dónde el principal objetivo es realizar una partición del proyecto en varias iteraciones planificadas y tratando de optimizarlas. Estas actividades son: Adaptación de la organización del proyecto, planeación, planeación de la iteración, planeación de los componentes específicos de la iteración, desarrollo de los componentes específicos, desarrollo de los componentes, preparación de la revisión y, revisiones de la iteración.

2.3.2.3. Rosenberg/Scott

Esta propuesta esta dividida en las siguientes fases: análisis de requerimientos, análisis y diseño preliminar, diseño e implementación. Las cuales involucran las siguientes actividades [RS1999]:

- Análisis de requerimientos:

Identificar dentro del dominio las clases y las relaciones de generalización y asociación existentes entre ellas, es decir, definir diagramas de clases a un alto nivel. Sin considerar aún los atributos u operaciones. En la figura 2.12a se muestra un diagrama de clases que muestra dos clases relacionadas con la clase uno a través de una relación de generalización y se observa que no se han definido atributos ni operaciones para las clases.

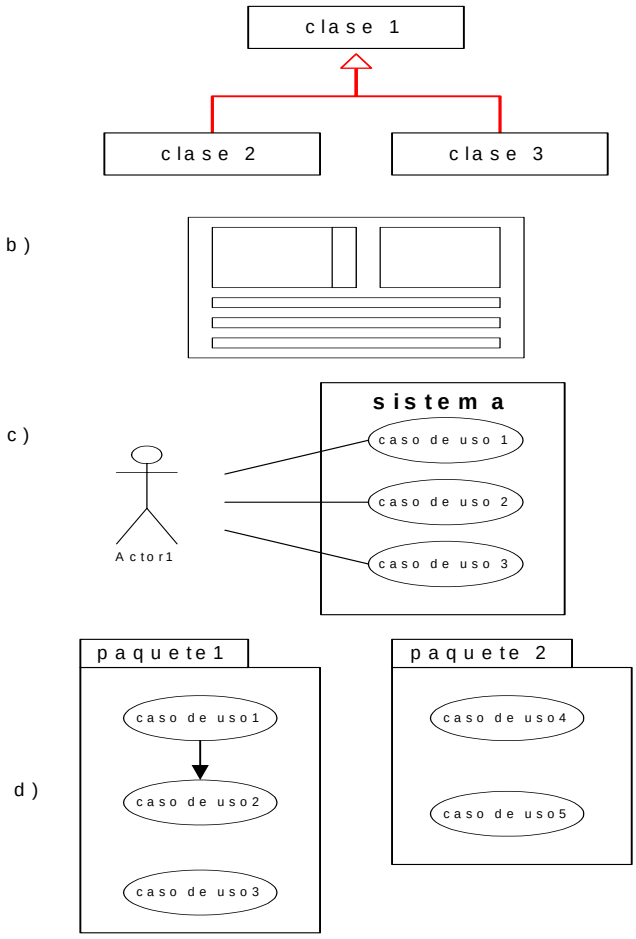


Figura 2.12. Modelación de Análisis de requerimientos (Rosenberg/Scott)

a) Realizar un prototipo básico del sistema propuesto y si es posible (y aplica) complementarlo con información del sistema existente. En este prototipo el objetivo es realizar un bosquejo general de lo que se espera realice el sistema y de cómo se integraría con el sistema anterior, si este existe. En la figura 2.12b se muestra un bosquejo de una interface.
b) Identificar los casos de uso y definir diagramas de casos de uso. (figura 2.12c)
c) Agrupar los casos de uso, utilizando diagramas de paquetes. Rosenberg/Scott sugieren agrupar los casos de uso en paquetes porque de esta forma se estaría estableciendo una división la lógica del sistema, además de que facilita la asignación de trabajo para el equipo de desarrollo. Como se muestra en la figura 2.12d, donde los casos de uso fueron agrupados en dos paquetes, cada paquete debería formar parte de un modulo del sistema.

Análisis y diseño preliminar:

a) Realizar descripciones textuales de los casos de uso. Una vez que se han modelado los diagramas de casos de uso, cada uno de estos deberá ser complementado con una explicación de lo que cada caso de uso significa (figura 2.15a).

b) Realizar un análisis de robustez que identifique para cada caso de uso escenarios y además que actualice los diagramas de clases. En la figura 2.13 se identifican los tres símbolos que se utilizan en el análisis de robustez:

   1. Objetos interface: los cuales usan los actores para comunicarse con el sistema.
   2. Entidades: que generalmente son objetos del dominio.
   3. Controles: que hacen la unión entre los objetos y las entidades.

Figura 2.13. Símbolos del Análisis de Robustez (Rosenberg/Scott)

El análisis de robustez se puede utilizar con cuatro propósitos diferentes:

1. Verificador: un análisis de robustez ayuda al desarrollador a tener la certeza que el texto del caso de uso es correcto y que no se tienen funcionalidades del sistema no especificadas o imposibles de realizar.

Los sustantivos utilizados en el texto descriptivo de los casos de uso deben corresponder con los nombres de los objetos utilizados en el análisis de robustez. Una práctica sugerida en [RS1999] es realizar en paralelo estas dos actividades.

2. Integridad: Ayuda a asegurar que se han tomado en cuenta todas las rutas alternas necesarias de acción en los casos de uso.

Debido a que gran parte del éxito en el proceso de una aplicación depende de los casos de uso, es relevante asegurar que se han escrito y realizado los diagramas adecuados [RS1999].

3. Identificación de objetos: si acaso no se hubieran identificado todos los objetos antes, el análisis de robustez resulta ser un filtro útil para descubrir objetos.

Esta característica es importante sobre todo para etapas posteriores, debido a que encontrar requerimientos de funcionalidad en etapas posteriores, como en el diseño puede resultar costoso [RS1999].

4. Diseño preliminar. Un análisis de robustez es la transición hacia el diseño, por lo que es importante definirlos claramente.

Especificaciones de los objetos del análisis de robustez

Para el desarrollo de un análisis de robustez es importante considerar algunas reglas (figura 2.14.):

1. Los actores pueden comunicarse solamente con objetos interface.
2. Los objetos interface pueden comunicarse con los actores y con los controles.
3. Las entidades pueden comunicarse solamente con los controles.
4. Los controles pueden comunicarse con otros controles y con objetos interface, pero no con los actores.

Lo más importante en la realización del análisis de robustez [RS1999] es: a) tener concordancia entre los sustantivos utilizados en los casos de uso con los nombres definidos para los símbolos, b) realizar una clara definición de éste análisis para que facilite la definición de diagramas de secuencia en la fase del diseño, c) no perder tiempo perfeccionando estos diagramas, si bien en el inciso anterior se afirma que la definición debe ser clara, esta deberá realizarse a un alto nivel y, d) seguir las reglas en la realización de los diagramas. En la figura 2.15b se muestra un diagrama del análisis de robustez, en el que un actor se relaciona por medio de un objeto interface con dos entidades, antes de que se relacione el objeto interface con las entidades se encuentran dos controladores.

c) Actualizar los diagramas de clases si es necesario, de manera que estos diagramas contengan todas las asociaciones y atributos identificados hasta este momento. Se utiliza el diagrama de clases definido en la fase anterior, pero se le agregan los atributos a las clases. La figura 2.14a corresponde al diagrama de clases de la figura 2.15c con la diferencia de que en esta última se incluyen los atributos identificados.

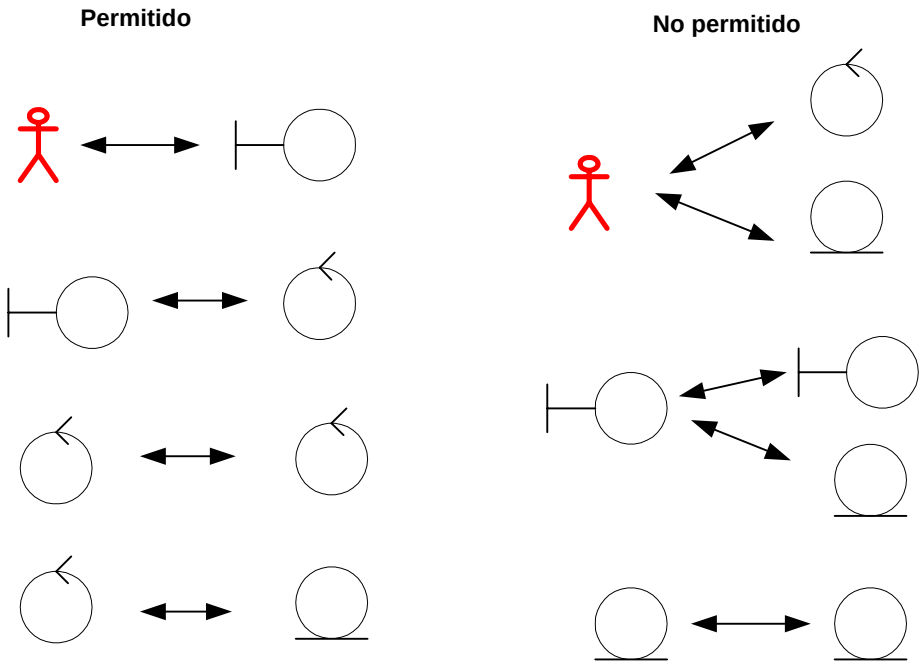


Figura 2.14. Reglas del Análisis de Robustez (Rosenberg/Scott)

- Diseño:

a) El diseño inicia identificando para cada caso de uso, los mensajes que deberán ser enviados entre los objetos y los métodos asociados utilizando diagramas de secuencia.

   En [RS1999] se sugiere realizar estos diagramas haciendo uso de los símbolos del análisis de robustez. Rosenberg/Scott afirman que al utilizar el análisis de robustez en combinación con los diagramas de secuencia se tiene la certeza de que los requerimientos del sistema están siendo considerados en la modelación del sistema. En la figura 2.15b se muestra un diagrama de análisis de robustez que da lugar al diagrama de secuencia mostrado en la figura 2.16a, en éste último diagrama se observan los mensajes generados entre los objetos. Los diagramas de secuencia deben ser coherentes con los diagramas del análisis de robustez tal como se observa con ambas figuras.

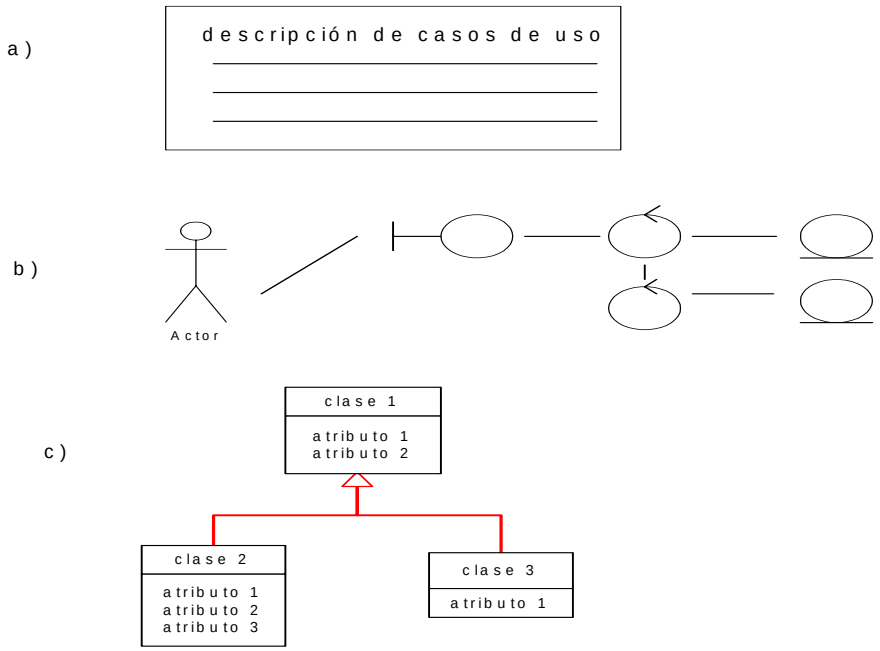


Figura 2.15. Modelación de análisis y diseño preliminar (Rosenberg/Scott)

b) Finalmente se siguen actualizando los diagramas de clase, donde ahora se le agregan los métodos identificados a casa clase. La figura 2.16b muestra el diagrama de clases modelado en 2.15c con los métodos ya incluidos, se observa también la adición de una cuarta clase, esto es válido debido a que Rosenberg/Scott proponen actualizar los diagramas de clase en cada fase, lo cual podría ser agregar fases o eliminar clases cuidando que todo el modelado sea consistente.

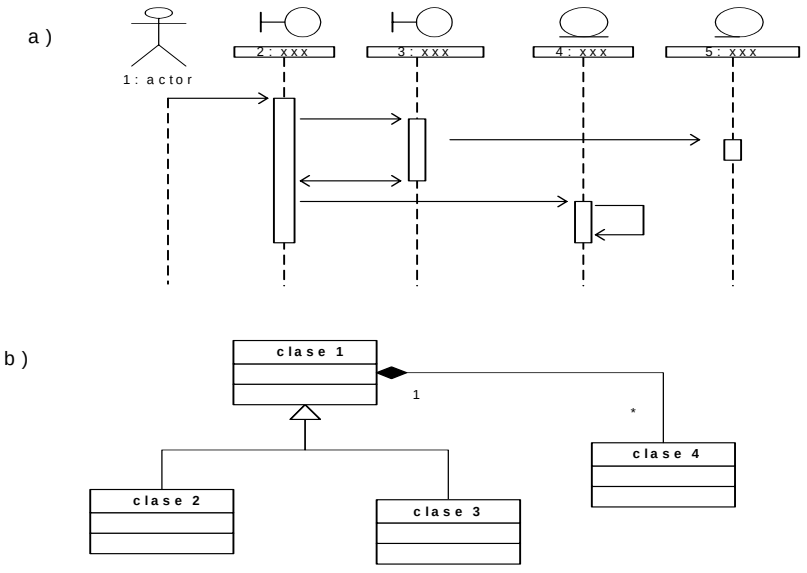


Figura 2.16. Modelación de Diseño (Rosenberg/Scott)

ï Implementación:

a) Modelar componentes si es necesario. Esto es agrupar las clases identificadas en piezas físicas del sistema, como se muestra en la figura 2.17, donde se observa cómo seis clases fueron asignadas a tres componentes, por ejemplo las clases órdenes de pedido y ventas se agrupan en el componente pc cliente.
b) Escribir el código en el lenguaje seleccionado.
c) Realizar pruebas de caja negra y pruebas de caja blanca.

En las pruebas de caja negra se proporcionan todas las posibles entradas al sistema, es por eso que los casos de uso son adecuados. El caso de uso representa lo que el desarrollador dijo que el sistema haría; si el sistema pasa la prueba que usa el caso de uso como escenario, entonces se puede afirmar que el sistema realmente realiza lo que se dijo que haría.

En las pruebas de caja blanca se examina la lógica interna. Estas pruebas analizan la estructura interna de los métodos que pertenecen a cada clase para confirmar que realicen lo que deben realizar. Para realizar estas pruebas se pueden utilizar las pruebas de unidad (son aquellas en las que se analiza paso a paso cada módulo) para verificar el funcionamiento de los métodos de cada clase.

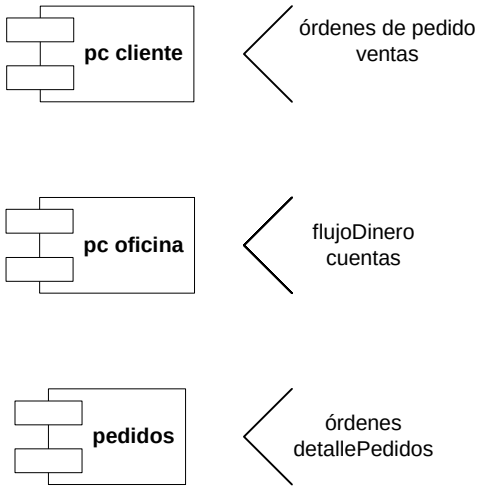


Figura 2.17. Modelación de componentes en la fase de Implementación (Rosenberg/Scott)

2.3.2.4. Conallen

Conallen realiza una propuesta de desarrollo para aplicaciones Web, definidas en [Con2000] como aplicaciones cliente/servidor las cuales tienen al menos en su arquitectura un navegador que comunicará a uno o más clientes por medio de un servidor Web y una aplicación servidor. Es relevante destacar que esta propuesta es específica para aplicaciones Web ya que algunos diagramas sugeridos son aplicables a la arquitectura, diseño e implementación de aplicaciones de este tipo. A continuación se detallan las fases definidas en [Con2000]:

- Especificación de requerimientos:

El principal objetivo de esta etapa es definir lo que el sistema debe de hacer, no cómo lo debe de hacer. La utilización de casos de uso y la descripción de escenarios en lenguaje natural (es decir la descripción de casos de uso), resultan útiles para esta especificación (figura 2.18).

En esta fase se realizan otras dos actividades: modelar los escenarios utilizando diagramas de secuencia y realizar el análisis de casos de uso con la finalidad de identificar los objetos que intervienen en los mismos (Rosenberg/Scott les llaman análisis de robustez, el cual es explicado en la sección anterior).

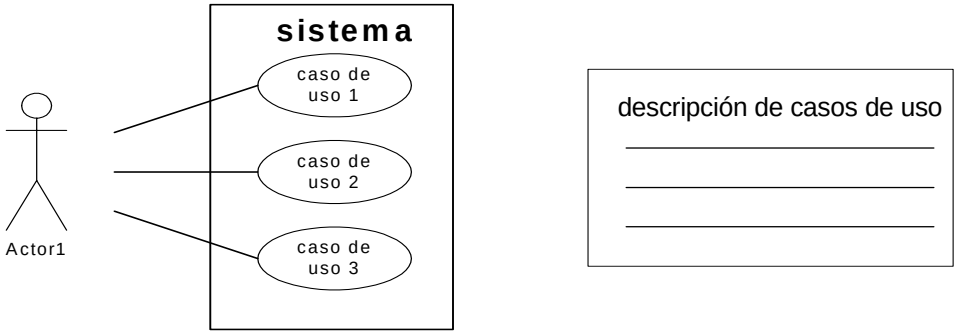


Figura 2.18. Modelación de especificación de Requerimientos (Conallen)

Los diagramas de secuencia se modelan utilizando el texto descriptivo de los casos de uso (escenarios) con el objetivo de representar la interacción entre los actores y el sistema, en esta fase estos diagramas se realizan a un alto nivel. La figura 2.19 muestra la interacción entre el actor y el sistema, para definir los mensajes entre ellos se utiliza la forma descriptiva del caso de uso.

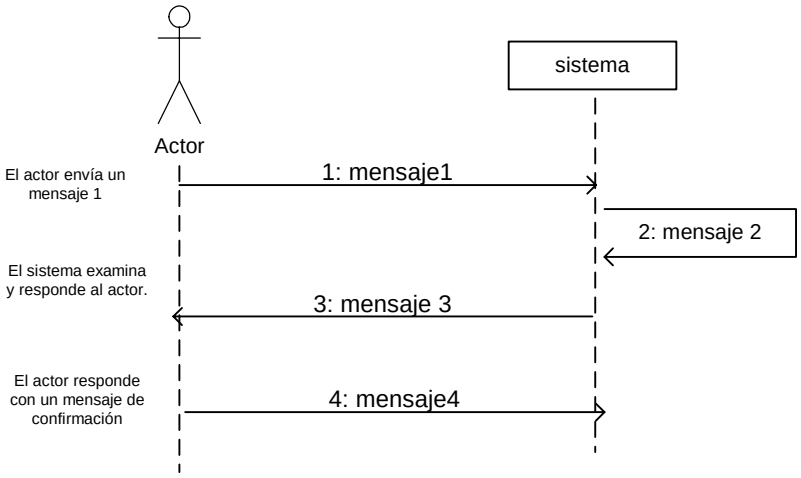


Figura 2.19. Modelación de diagramas de secuencia en la fase de Requerimientos (Conallen)

ï Análisis:

El análisis es el proceso de examinar los requerimientos y realizar un modelo conceptual del sistema. En esta etapa se detallan las clases y la colaboración entre ellas (como se muestra en la figura 2.20), diagramas de secuencia, diagramas de colaboración y diagramas de actividad.

Los diagramas de secuencia son utilizados para representar la relación entre los casos de uso y la estructura de las clases. En esta fase los diagramas de secuencia hacen uso del análisis de casos de uso realizado en la especificación de los requerimientos. La figura 2.21 muestra la secuencia entre el actor y los objetos identificados en el análisis de casos de uso, la descripción de los casos de uso definen los mensajes entre ellos.

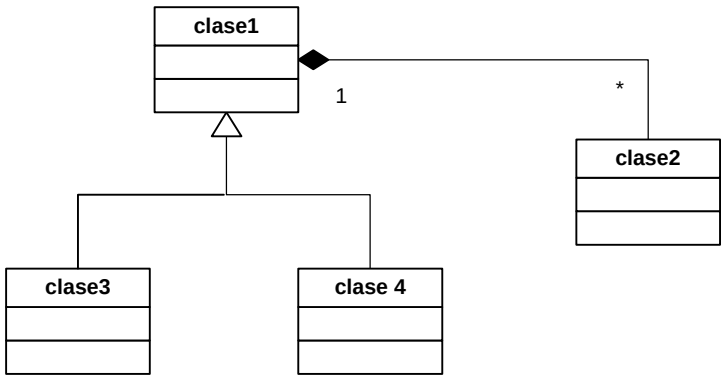


Figura 2.20. Modelación de diagramas de clase en la fase del Análisis (Conallen)

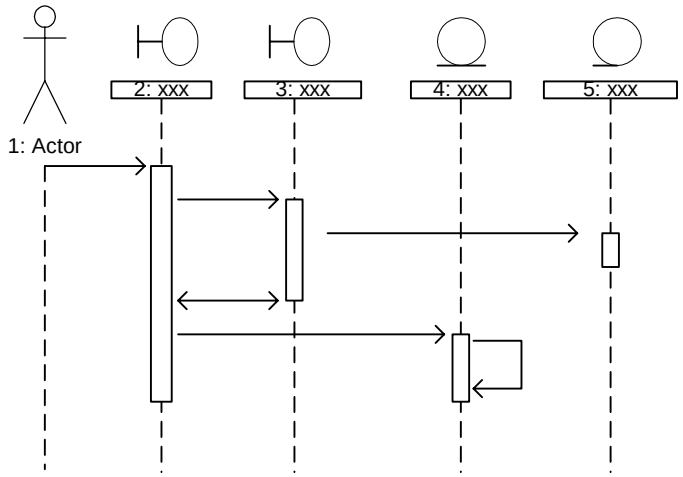


Figura 2.21. Modelación de diagramas de secuencia en la fase del Análisis (Conallen)

Básicamente los diagramas de colaboración y los de secuencia son lo mismo, la diferencia es que los diagramas de secuencia se enfocan en el tiempo y los de colaboración en la organización de los objetos [Con2000]. Conallen representa estos diagramas haciendo uso de los objetos identificados en el análisis de casos de uso, donde los objetos son localizados en cualquier lugar del diagrama y los mensajes son enviados de un objeto a otro, los cuales son numerados para representar el orden en que estos deben ser enviados como se muestra en la figura 2.22.

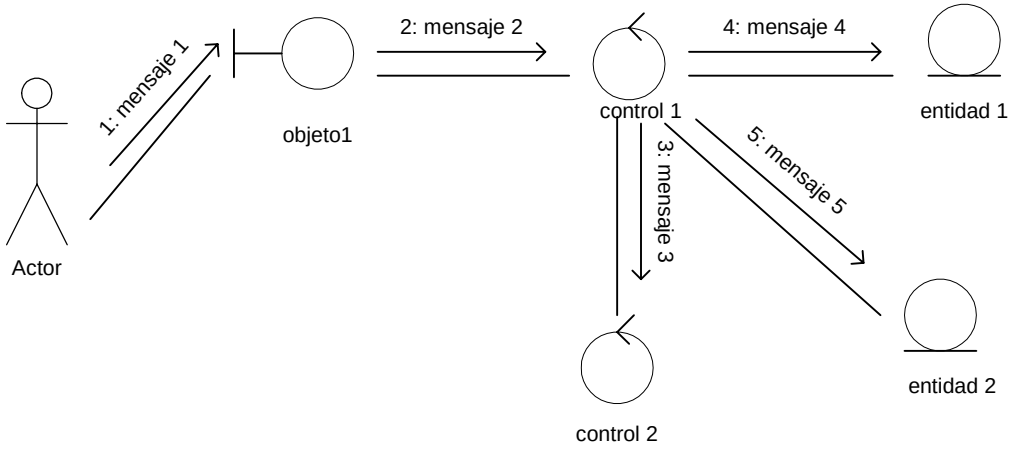


Figura 2.22. Modelación de diagramas de colaboración en la fase del Análisis (Conallen)

Conallen utiliza los diagramas de actividad para representar las actividades de una operación específica. La figura 2.23 muestra un diagrama de actividad de una operación en donde se muestra el flujo de control de una actividad a otra. Para modelar una operación se requiere de identificar los parámetros y las precondiciones del estado inicial y las poscondiciones al estado final.

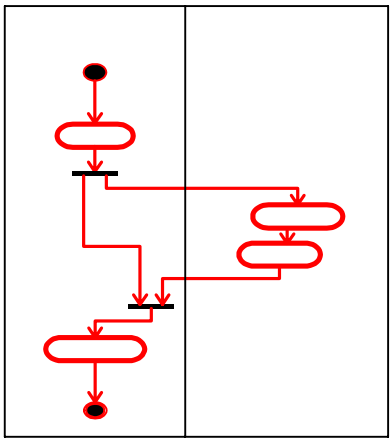


Figura 2.23. Modelación de diagramas de actividad en la fase del Análisis (Conallen)

- Diseño:

Se utilizan los diagramas producidos durante el análisis y además se modela la arquitectura. En esta fase se define una arquitectura de software que sea capaz de cubrir todos los requerimientos del sistema y de realizar los casos de uso. Conallen define la arquitectura de aplicaciones Web en [Con2000], utilizando patrones de diseño (recordemos que esta es una metodología para aplicaciones Web, la cual es definida por Conallen como un sistema cliente/servidor que tiene al menos un navegador que comunicará a los clientes con el servidor Web y una aplicación servidor que administrará la lógica del negocio –los procesos-). Los patrones definidos por Conallen para diseñar la arquitectura son:

1. Cliente liviano. Usado principalmente para aplicaciones Web, en donde se requiere de poco control por parte del cliente. El cliente requiere de formatos simples. Y la lógica del negocio es ejecutada del lado del servidor. (la lógica del negocio se refiere a los procesos)
2. Cliente pesado. Gran parte de la lógica del negocio es ejecutada del lado del cliente.
3. Clientes distribuidos. Es utilizado para sistemas distribuidos, el navegador es utilizado como dispositivo de envío y recepción.

El diseño incluye dos actividades [Con2000]:

1. Particionar el sistema determinando los objetos del cliente y los del servidor.
2. Definir y separar las interfaces de usuario y las páginas Web.

Para el diseño de aplicaciones Web, como ya se mencionó, la ubicación de los objetos que se implementarán en el cliente y en el servidor, es importante y está en función de los patrones de arquitectura de software definidos [Con2000]:

1. Cliente liviano: la ubicación de todos los objetos será en el servidor.

2. Cliente pesado: una forma fácil de definir cuales objetos estarán del lado del cliente y cuales del lado del servidor, es identificando aquellos objetos que no pueden estar en el servidor y entonces ubicarlos en el cliente.
3. Cliente distribuido: para hacer una correcta división de los objetos de un sistema distribuido, el mejor criterio es identificar en que parte los objetos tienen mayor acceso a los datos y en donde se requiere mayor rapidez de procesamiento. Esta identificación servirá para dividir los objetos, además se debe considerar en una arquitectura de este tipo que la partición de objetos depende de la naturaleza misma de los objetos individuales.

Dado que el objetivo de este documento es analizar la metodología propuesta por cada autor y no los detalles de las aplicaciones, no se profundarizará en el modelado de estos patrones (la referencia completa se encuentra en [Con2000]). La idea principal de esta fase según Conallen es utilizar los diagramas generados en la fase del análisis y modelar la arquitectura, de modo que la transición hacia la codificación sea un paso prácticamente automático.

- Codificación:

Las actividades de la codificación son: mapeo del diseño en código y componentes, realización de pruebas unitarias y modificación en cualquiera de los diagramas si ocurren cambios en el código que repercutan en ellos.

- Prueba:

Se sugiere la aplicación de las siguientes pruebas:

1. Pruebas unitarias: para verificar el funcionamiento de pequeñas partes del sistema, cada programador es responsable de aplicarlas.
2. Pruebas de integración: estas pruebas validan las interfaces de manera individual y la facilidad de integración con otras.
3. Pruebas del sistema: se verifica que todos los requerimientos hayan sido satisfechos.
4. Pruebas de aceptación: realizada por los usuarios finales del sistema.

2.4 Resumen

- Web Services es la última generación en la integración de aplicaciones. Un Web Service es la interface que describe una colección de operaciones que están disponibles en la red, mediante el uso de estándares de protocolos. Estos son SOAP para comunicar Web Services, UDDI para registrar y publicar servicios y WSDL para describir los servicios; estos estándares forman el núcleo de los Web Services.

- Las características principales de los Web Services son: independencia de plataforma, interoperabilidad y la utilización de estándares abiertos.

- La arquitectura orientada a servicios, se basa en tales estándares y además se identifican tres entidades principales: proveedores de servicios, repositorios de servicios y solicitadores de servicios. Entre estas entidades se generan tres operaciones básicas: búsqueda de servicio, publicación de servicio y comunicación.

- UML es un lenguaje utilizado para modelar sistemas, pero por si mismo no representa un proceso de desarrollo.

- Larman, Oestereich, Rosenberg/Scott y Conallen son algunos autores quienes proponen metodologías de desarrollo utilizando UML para modelar cada fase.

En este capítulo se ha discutido acerca de por qué Web Services es relevante, se ha mostrado un panorama general de UML y se han presentado metodologías de cuatro diferentes autores.

A continuación se presenta una tabla en la que se identifican los requerimientos de Web Services y la representación de la arquitectura mostrados en este capítulo en la sección 2.1.4 y la sección 2.1.5 respectivamente, haciendo una relación con lo que cada autor (Conallen [Con2000], Oestereich [Oes1999], Larman [Lar1999], Roserberg/Scott [RS1999]) aportaría para esta representación.

	Larman	Oestereich	Rosenberg/Scott	Conallen
integración (representación de interfaces)	No aplica	Si aplica	No aplica	No aplica
Tecnológicos (representación para estándares o tecnologías utilizadas)	No aplica	No aplica	No aplica	Si aplica
Orientados al servicio (seguridad y confiabilidad)	No aplica	No aplica	No aplica	Si aplica
representación de la arquitectura	Si aplica	Si aplica	No aplica	Si aplica

Tabla 1. Web Services y metodologías

Como resultado del análisis presentado en este capítulo y la relación mostrada en la tabla anterior se observa que a excepción de Conallen [Con2000] ningún otro autor proporciona modelaciones para aplicaciones distribuidas como es el caso de Web Services, sin embargo la propuesta de Conallen está orientada a aplicaciones Web. Por lo que se observa la necesidad de proponer un proceso de desarrollo para la integración de aplicaciones orientada a Web Services. En el siguiente capítulo se presenta un proceso iterativo utilizando un conjunto de diagramas que está basado en las diferentes propuestas presentadas en este capítulo y que guiará a los desarrolladores en cada fase.

Capítulo 3 Propuesta
Proceso de desarrollo para Web Services

En este capítulo se propone un proceso de desarrollo de Web Services, para lo que sugerimos la elaboración de dos documentos iniciales y una serie de actividades ordenadas y divididas en seis fases para desarrollar Web Services. Esta propuesta sugiere la utilización de un proceso, mediante el cual identificamos qué, cómo y cuando. El qué hacer se responde mediante la identificación de los artefactos para cada fase de desarrollo, el cómo se satisface mediante sugerencias de aplicación de los mismos y el cuándo queda definido por cada fase. Se definen también, los conceptos relacionados, tales como proceso, artefacto, iterativo y actividad. Este capítulo tiene dos secciones en la primera parte es una introducción al proceso y en la segunda se desarrolla detalladamente el proceso propuesto orientado a Web Services.

3.1. Introducción a un proceso de desarrollo

En esta sección se definen los conceptos importantes dentro del contexto de este capítulo y de la propuesta realizada.

Un *proceso de desarrollo de software* como lo define Larman en [Lar1999] “es una forma de organizar las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de software”. Es importante en este punto destacar la diferencia entre metodología y proceso. Una *metodología* abarca los pasos individuales y secuenciales del desarrollo, proporcionando una guía específica y detallada de los mismos, un *proceso* es más abstracto que una metodología pues no detalla tan cuidadosamente todos los aspectos a los que se aplica, por lo cual es más utilizado en cuestiones más genéricas del desarrollo.

El proceso que se propone se rige por un desarrollo iterativo dividido en fases, que sugieren una serie de actividades e identificando los artefactos para cada fase. Ampliando este concepto se define:

- *Desarrollo iterativo*: se basa en el perfeccionamiento secuencial a través de varios ciclos de desarrollo. Cada iteración involucra una serie de actividades, tales como requerimientos, diseño, implementación o alguna otra fase definida.

Mediante un proceso iterativo un proyecto puede incorporar nuevas funcionalidades y requerimientos de manera controlada. Algunas de las ventajas son:

1. Identificación de riesgos desde las primeras fases de desarrollo
2. Control de cambios
3. Los requerimientos son ajustables a las necesidades del proyecto.

Se genera retroalimentación en las fases iniciales.

- *Fase*: es cada una de las etapas en las que se divide el proceso, esta formada por una serie de actividades que producen un resultado observable. Ejemplos de fases son: análisis, definición de arquitectura entre otros.

- *Actividad*: define el trabajo que se tiene que realizar, es una unidad de trabajo que produce un resultado importante en el contexto del desarrollo [Kru2000]. Una actividad puede ser realizada varias veces sobre el mismo artefacto, esto es común cuando hablamos de procesos iterativos. Ejemplos de actividad son: identificar actores y casos de uso, realizar un diagrama específico, entre otros.

- *Artefacto*: es una pieza de información que es producida, modificada o usada por un proceso [Kru2000]. En otras palabras los artefactos son los entregables de cada fase, es decir, es lo que el proceso produce o usa mientras se está desarrollando. Un artefacto es usado como entrada para desarrollar una actividad y son el resultado de tales actividades. Ejemplos de artefactos son: diagramas de casos de uso, resultados de pruebas, entre otros.

3.2. Proceso de desarrollo propuesto

Esta sección presenta a detalle el proceso, definiendo los artefactos que deben realizarse en cada fase y especificaciones para la elaboración de los documentos previos sugeridos. Los cuales son:

Documentos previos:

- Introducción
- Panorama general

Fases:

- Requerimientos
- Definición de arquitectura
- Análisis
- Diseño
- Codificación e implementación
- Pruebas

En las siguientes secciones se describen a detalle las actividades a realizar y los artefactos a producir para cada fase de desarrollo.

3.2.1. Documentos Previos

3.2.1.1. Introducción

En la introducción básicamente se plantea la problemática, se define una solución y se identifican los beneficios. Este documento es importante porque identifica si es aplicable utilizar Web Services como medio de integración o bien si se podría utilizar algún otro medio; lo cual queda fuera del contexto de este trabajo. Sin embargo es importante identificar si Web Services es la solución adecuada para la problemática planteada y esta actividad es una buena práctica para tomar esta decisión.

3.2.1.2. Panorama General

En la introducción básicamente se plantea la problemática, se define una solución y se identifican los beneficios. Este documento es importante porque identifica si es aplicable utilizar Web Services como medio de integración o bien si se podría utilizar algún otro medio; lo cual queda fuera del contexto de este trabajo. Sin embargo es importante identificar si Web Services es la solución adecuada para la problemática planteada y esta actividad es una buena práctica para tomar esta decisión.

3.2.2 . Fases

3.2.2.1. Requerimientos

De acuerdo a la clasificación de los requerimientos mencionada en el capítulo 2, se tienen tres tipos de requerimientos:

a) Requerimientos tecnológicos

Los requerimientos tecnológicos serán satisfechos por los estándares propios del desarrollo de Web Services. A continuación se mencionan los estándares utilizados.

Este documento debe especificar los estándares utilizados, describiendo las versiones a las que pertenecen y aspectos técnicos que se deberían considerar en la implementación de los mismos.

1. Protocolo: SOAP
SOAP permitirá que Web Services desarrollados en diferentes plataformas puedan comunicarse mediante transferencia de mensajes, usando http y XML como mecanismos de intercambio de información.

2. Registro y descubrimientos de servicios: UDDI
Mediante la forma unificada y sistematizada proporcionada por UDDI, se definirá la búsqueda de proveedores de servicios y el registro de un servicio.

Los servicios pueden ser públicos o privados (ver sección 1.5 del capítulo 2), por lo que en este documento debe especificarse el tipo de servicio que se desea publicar y el tipo de servicios a los que se desea accesar. Es relevante recordar como se mencionó en el capítulo 2, que los servicios públicos actualmente son más explotados que los privados, ya que los servicios privados requieren de una mayor programación y más allá de eso, no se tiene la certeza de que el servicio pudiera estar disponible. Mientras que los servicios públicos tienen una ubicación definida y se tiene la certeza de que el servicio estará disponible en cualquier momento.

3. Descripción del servicio: WSDL
Con WSDL se describirá la interface y proveerá a los Web Services de un punto de contacto. WSDL proporcionará básicamente la información al respecto de la descripción del servicio a nivel aplicación y detalles específicos de la interacción entre Web Services.

b) Requerimientos de integración

Los Web Services desearán compartir sus servicios; para hacerlo será necesario definir interfaces donde se definan.

La compartición de servicios mediante interfaces, puede clasificarse de la siguiente manera:

a) Respuesta genérica a diferentes solicitudes: Ocurre cuando un Web Service recibe solicitudes de servicio por parte de diferentes Web Services y la respuesta puede ser la misma para todos. Es decir quizás la solicitud no implique el envío de datos que pudieran hacer variar la respuesta del Web Service proveedor, sino que simplemente responderá con el servicio. En la figura 3.1 el Web Service proveedor recibe solicitudes desde tres Web Services cliente diferentes, sin embargo el servicio es implementado en una misma interface mediante la cual se realiza la compartición.

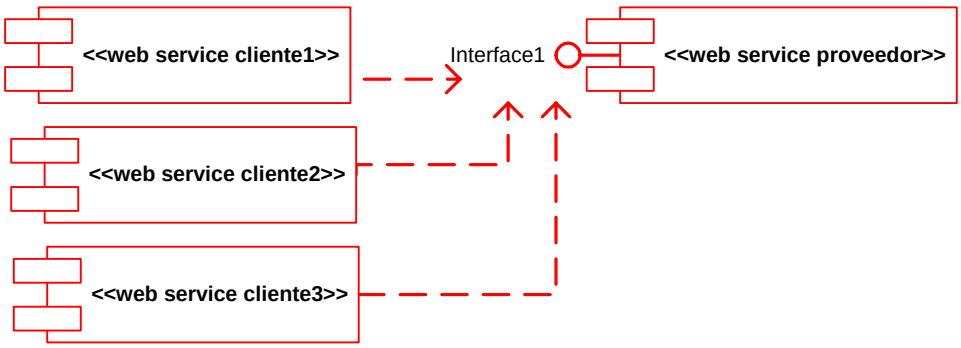


Figura 3.1. Respuesta genérica a diferentes solicitudes

b) Una solicitud una respuesta: cuando un Web Service recibe solicitudes específicas por parte de diferentes Web Services, por lo que su respuesta debe ser también específica; es decir la compartición es uno a uno. En este caso quizá la solicitud envía datos que harán que la respuesta pueda variar. En la figura 3.2 se muestra que el Web Service proveedor satisface peticiones de servicio a tres Web Services cliente, pero dependiendo de sus peticiones el Web Service proveedor responde usando tres interfaces que implementan diferentes servicios.

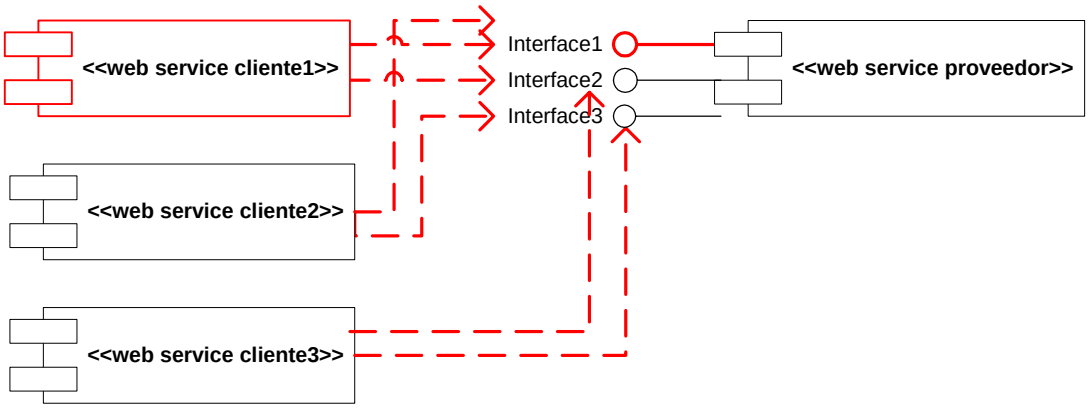


Figura 3.2. Una solicitud-una respuesta

c) Requerimientos orientados al servicio

A continuación se describen los requerimientos que serán satisfechos por las especificaciones propias de los estándares utilizados.

1. Seguridad

Son los aspectos necesarios para garantizar el envío de mensajes, la transmisión de información y la fiabilidad de los servicios, los cuales son cubiertos por los estándares.

Cuatro de los requerimientos de seguridad: confidencialidad, autenticación, integridad de los datos y prueba de origen pueden ser satisfechos mediante la encriptación (protección a la comunicación de los datos, además de restringir el acceso a las aplicaciones). La autorización (otro requerimiento de seguridad) se satisface mediante el intercambio de mensajes de SOAP.

Un modelo de seguridad que puede ser implementado es el propuesto por Oellerman en [Oell2001], el cual consiste de contestar básicamente tres cuestiones:

1. ¿Quién usará el Web Service?, es decir estará disponible en una Intranet o en la Web. La diferencia radica en que si estará disponible en una red local (Intranet) se tendrán perfectamente identificados los servicios (servicios privados), si es en la Web habrá que decir si son servicios públicos o privados. Lo que da lugar a la siguiente cuestión.
2. ¿Los servicios serán públicos o privados?
3. ¿Cuál modelo de seguridad será implementado? Para contestar esta pregunta es necesario haber decidido si el Web Service, será implementado en una red local o si estará disponible en el Web. En caso de que sea esto último una alternativa de solución es la implementación de un servidor

proxy, mediante el cual se puede garantizar que las solicitudes a los servicios serán direccionadas a las aplicaciones correctamente.

Este modelo de seguridad requiere de:

a) Una aplicación de autenticación. El objetivo es proporcionar una adecuada funcionalidad a los usuarios y garantizar que sólo tendrán acceso los usuarios a los que les ha sido permitido. Esto requiere un mecanismo de autenticación, el más común es la asignación de una cuenta de usuario y una contraseña.
b) Integridad en el envío de datos, para lo cual una alternativa es la encriptación de datos, mediante Secure Sockets Layer (SSL) que es la técnica mas utilizada para encriptar datos, sin embargo se podrían utilizar algunas otras tales como Kerberos, Smart Cards, etc. (Definir la seguridad en un Web Services, es uno de los tópicos en esta área que no se encuentran definidos completamente, por otro lado la toma de decisión acerca de cual técnica utilizar esta fuera del contexto de esta tesis).

2. Confiabilidad

Este requerimiento será satisfecho por la implementación propia del Web Service, lo relevante es considerar dentro de la fase de codificación si los servicios serán privados o públicos (ver Capítulo 2, sección 1.5) y tener alternativa en caso de que el servicio pudiera no estar disponible para el caso de los servicios públicos.

Identificación de los actores y los casos de uso

Los casos de uso son una forma de identificar y expresar la interacción entre los usuarios del sistema y el sistema. Un caso de uso es una secuencia de acciones que un actor realiza dentro del sistema. En el caso concreto de Web Services, los actores son las aplicaciones (Web Services clientes que participan en la integración) que solicitarán servicios y los casos de uso deberán estar en función de los servicios ofrecidos por el Web Service proveedor.

3.2.2.2. Diagramas de casos de uso

Los casos de uso se modelan mediante diagramas que representan las relaciones entre los actores y diferentes casos de uso. En la figura 3.3 se muestra un ejemplo típico de Web Services, donde una aplicación que desea un servicio deberá buscarlo dentro de los servicios publicados por el Web Services proveedor.

Cuando se tienen una gran cantidad de casos de uso, generalmente se tiende a agruparlos en base a su funcionalidad, sin embargo esto es un error [Oes1999], el propósito de los casos de uso es simplemente representar interacciones entre el sistema y los actores; los casos de uso no son requerimientos ni especificaciones funcionales [Lar1999].

Otro error común en el modelado de los casos de uso es representar pasos, operaciones o transacciones individuales como casos de uso [Lar1999]. Por lo que se debe aclarar que un caso de uso es una descripción de un proceso de principio a fin que implica generalmente más de una actividad.

Asociaciones entre casos de uso

Dentro de los diagramas de casos de uso es posible, identificar algunas relaciones entre ellos, estas deben ser representadas utilizando <<include>> y <<extend>>. La mayoría de las asociaciones entre los casos de uso en un diagrama simplemente implican que uno invoca a otro [Con1999]. Sin embargo no se debe invertir demasiado tiempo en identificar tales relaciones y una forma de identificar la diferencia entre ellos es manteniendo en mente la idea de que invocar es similar a include y preceder a extend.

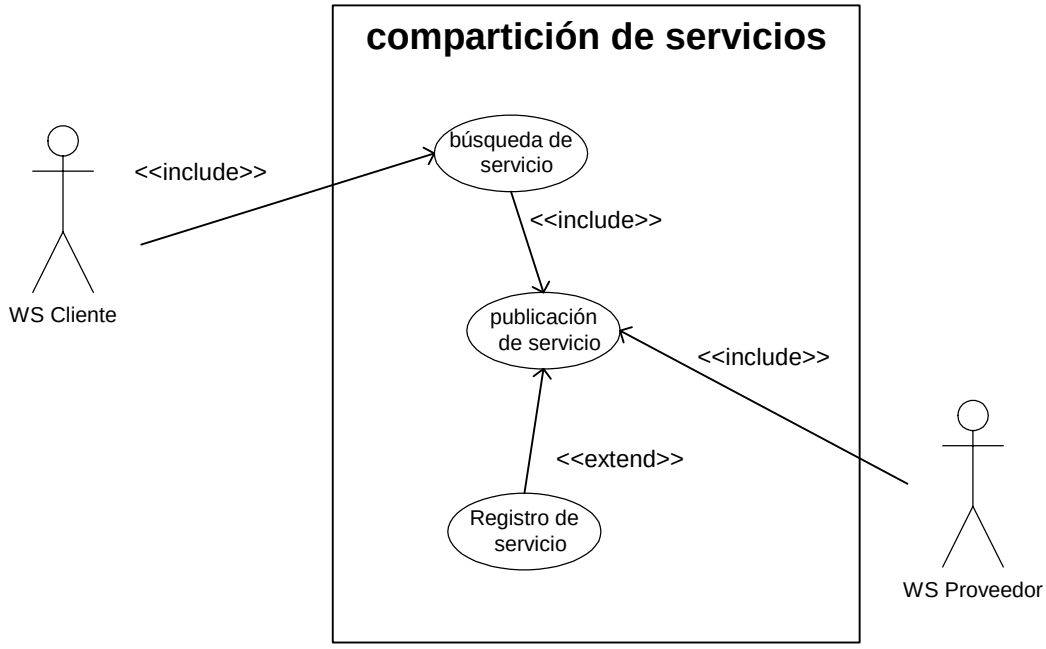


Figura 3.3. Diagrama de caso de uso para la compartición de servicio

En la figura 3.3 podemos ver el uso de include y extend; WS Cliente realiza el caso de uso búsqueda de servicio, que a su vez invocará al caso de uso publicación de servicio el cual es realizado por WS Proveedor. Por otro lado este último caso de uso requiere de realizar antes el registro del servicio.

Casos de uso agrupados en paquetes

Ya se ha comentado que agrupar los casos de uso en base a su funcionalidad no es una buena práctica. No obstante se pueden agrupar en base a una clasificación técnica, para lo cual se utilizan paquetes. Esto es útil debido a que en las primeras etapas del proceso iterativo es difícil tener casos de uso definidos completamente. Realizar esta agrupación representa un inicio hacia la modularización y por otro lado, permite distribuir el trabajo entre el equipo de desarrollo [Con2000].

Recomendaciones para especificación de requerimientos

- Cada requerimiento debe ser claro y conciso, evitando descripciones que puedan ser interpretadas de diferentes formas.
- Cada requerimiento debe enfocarse a puntos específicos del sistema, es más fácil así verificar si se está satisfaciendo.
- Todos los requerimientos deben ser verificables, evitando especificaciones con adjetivos que resulten ambiguos y/o cualitativos.

Estas tres recomendaciones fueron extraídas de la propuesta presentada por Conallen [Con2000].

Requerimientos y Casos de uso

- Un caso de uso describe partes del funcionamiento del sistema [Oes1999].
- Un requerimiento describe un aspecto que define el funcionamiento esperado [Lar1999].
- Un caso de uso puede satisfacer uno o más requerimientos funcionales [Oes1999].
- Un requerimiento funcional puede ser satisfecho por uno o más casos de uso [Oes1999].

3.2.2.3. Descripción de los casos de uso

La manera precisa de expresar los requerimientos de la interacción entre el sistema y un actor es hacerlo a través de casos de uso tal como lo dice Conallen en [Con1999]. A través de esta técnica se obtiene información más detallada de cómo el sistema debe funcionar y cómo responder a entradas externas por parte de un actor.

En el contexto de Web Services los casos de uso expresarán la interacción Web Services (actor) y los servicios a los que deseen accesar (caso de uso).

Los casos de uso contienen una descripción narrativa del uso de un escenario específico (situaciones en las que operará el sistema), tal que un actor proporcionará entradas para que el sistema procese y reporte resultados [Con1999]. Un caso de uso podría contener la descripción de más de un escenario; sin embargo, solo existirá un escenario principal y los otros escenarios son llamados como escenarios alternativos o rutas alternativas.

Los casos de uso deben ser escritos en el lenguaje propio del sistema [Con1999]. De manera que los usuarios y el equipo de desarrollo los puedan entender correctamente. Pueden tener diferente formato y longitud; sin embargo la manera como Conallen sugiere en [Con1999] permite establecer claramente el objetivo y la forma de realizarla. Y en base a esta sugerencia se propone describir los casos de uso para Web Services. Se complementa además con algunas sugerencias realizadas por Larman y Oestereich.

La información que debe especificarse en los casos de uso es:

- ID único. Idealmente un número automático que identifique a cada caso de uso de manera particular.
- Nombre. Es el título que identificará a cada caso de uso.
- Autores. Nombre de los miembros del equipo quienes están realizado cada caso de uso.
- Precondiciones. Es una descripción textual de las condiciones que deben ser identificadas antes de que un caso de uso sea representado.
- Poscondiciones. Es una descripción textual del estado del sistema después de que se ha realizado un caso de uso
- Descripción. Es la narración de lo que las actividades que el autor realiza, describiendo paso a paso el proceso de ellas.

(Hasta estas especificaciones son las basadas en la propuesta de Conallen.)

- Versión. Larman sugiere [Lar1999] asignar los casos de uso, refiriéndose al hecho de clasificarlos mediante un identificador en relación a la versión. Esta práctica resulta útil, debido a que proponemos un proceso iterativo, tener la referencia de la versión (haciendo referencia al número de iteración en que se realiza cada caso de uso) ayuda a los miembros del equipo a tener un mejor control sobre versiones y modificaciones.

- Actores y Descripción. Además es relevante tener claramente identificado el actor o los actores que intervienen en el caso de uso y desde luego la descripción narrativa de cada caso de uso, tal como lo propone Oestereich [Oes1999].

En la figura 3.4 se presenta una plantilla para la especificación de la descripción de los casos de uso.

Errores y recomendaciones en la descripción de los casos de uso

- Realizar una descripción breve de los casos de uso, que no abarque los aspectos relevantes (identificados durante la validación de la propuesta).
- Describir interacciones, ignorando las respuestas del sistema [Lar1999].
- Definir como casos de uso, los pasos que forman parte de un proceso completo de un caso de uso en particular [Lar1999].
- Describir casos de uso en voz activa y no pasiva [RS1999].

Recomendaciones

- No considerar las precondiciones o poscondiciones del caso de uso [Oes1999].
- No considerar todas las alternativas para un caso de uso [RS1999].
- No invertir tiempo en definir si se debe utilizar las relaciones include o extend utilizadas en UML.
- Agrupar casos de uso en base a su funcionalidad [Oes1999].

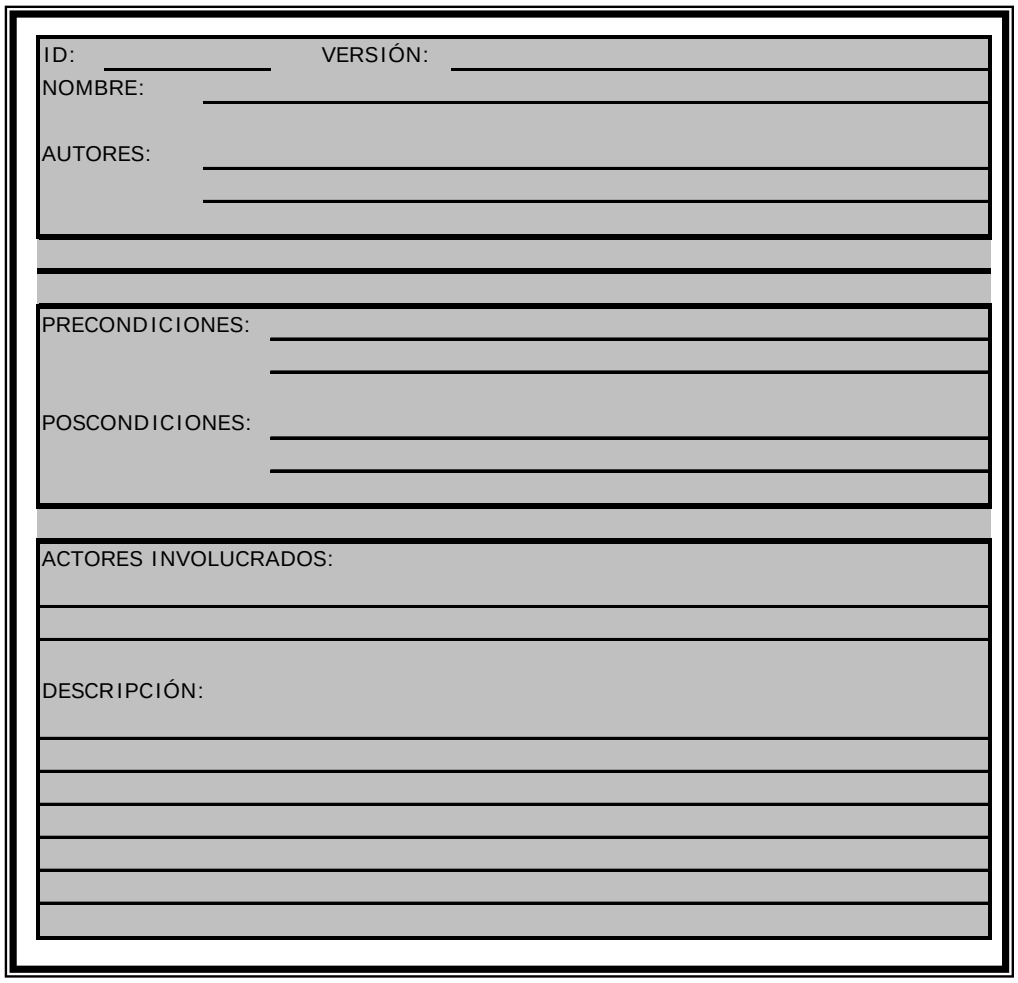


Figura 3.4. Plantilla para la descripción de los casos de uso

3.2.2.4. Análisis de robustez

Este concepto fue introducido por Jacobson en 1991, consiste en analizar el texto descriptivo de cada caso de uso para: a) identificar los objetos que participarán en el flujo de eventos de los casos de uso y, b) identificar las responsabilidades, atributos y asociaciones de las clases. Estos objetos se clasifican en tres tipos:

- Objetos interface. Representan la interface entre los actores y el sistema.
- Entidades. Objetos descritos en los casos de uso que forman parte del modelo del dominio.
- Controles. Representan los procesos, es decir la unión entre objetos interface y entidades.

Rosenberg/Scott destacan la importancia de este análisis en [RS1999], argumentado que esta técnica representa la transición entre el análisis (qué hacer) y el diseño (cómo hacerlo). Aunque este análisis no forma parte propiamente de UML, sí aparece como extensión. Para Conallen este tipo de análisis solo identifica los objetos que se utilizarán en los casos de uso, mientras que realizar un análisis de robustez forma parte importante y fundamental en la propuesta presentada por Rosenberg/Scott [RS1999].

Este análisis utiliza la descripción de los casos de uso, de donde se identifican los objetos interface, los actores, las entidades y los controles que cada caso de uso utiliza. Y se rige por las reglas mencionadas en el capítulo 2, sección 2.3.2.3. Es importante aclarar que deberá existir un diagrama de este tipo por cada caso de uso, independientemente de la agrupación que haya sido realizada.

En la figura 3.5 se muestra un diagrama para el caso de uso de búsqueda de servicio. En este ejemplo un solo actor esta involucrado, se observa también que se enlaza a otro caso de uso, sin embargo no se modela éste último; ya que requiere de su propio diagrama. En el ejemplo mostrado el actor se relaciona mediante el objeto interface ventana de registra, el registro es validado y despliega la lista de servicios que se tienen disponibles, para lo cual el sistema busca en los servicios publicados.

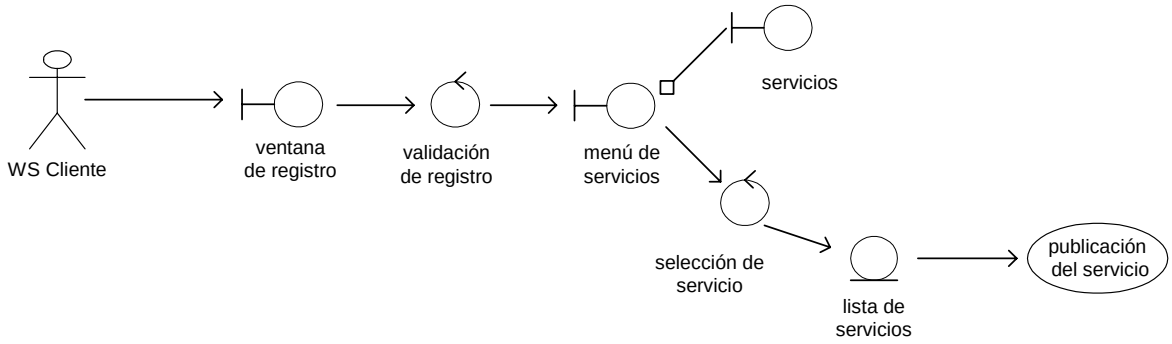


Figura 3.5. Diagrama del análisis de robustez para la búsqueda de servicios

La relevancia de este tipo de análisis es que representa una transición hacia la etapa del diseño, aunque en este momento la fase siguiente sea el análisis. Realizar este tipo de análisis permite describir los casos de uso que no hayan sido descritos completamente, ya que este análisis resulta un filtro por estar basado en la descripción de los casos de uso. Por otro lado permite identificar los objetos del sistema.

Beneficios del análisis de robustez

- Identifica errores de definición y escritura en los casos de uso.
- Ayuda a escribir los casos de uso correctamente en voz activa.
- Facilita la identificación de lo que el sistema hace (Casos de uso) con la forma como el sistema debe trabajar (diagramas de secuencia).
- Proporciona un lazo de conexión entre el análisis y el diseño.
- En esta fase resultan sencillos de interpretar y de dibujar en comparación con otros diagramas tales como diagramas de secuencia.

Estos cinco beneficios son extraídos de la propuesta de [RS1999] y se comprobó durante la validación que son aplicables al desarrollo de Web Services.

3.2.2.5. Diccionario del dominio

Mediante este documento técnico se definirán los términos propios del dominio del problema (contexto del ambiente sobre el que se desarrolla el Web Service) y que son fundamentales para la comprensión del Web Services que se está desarrollando. El diccionario de dominio se inicia en esta etapa, y se va incrementando el número de términos incluidos en él, conforme se vayan identificando nuevos, en fases posteriores.

El objetivo de este documento es tener un registro de la descripción de las reglas del dominio, es decir identificar:

- Conceptos que son usados en el sistema (documentos, personas, aplicaciones, etc.) y su significado.
- Propiedades, es decir, datos que requiere y datos que proporciona, describiéndolos brevemente.
- Rol que desempeña, es decir si es un caso de uso, es un actor, es un atributo de alguna clase, una operación, etc.
- Instrucciones o acciones que realizará y eventos que puedan existir entre una acción y otra.

Algunas otras especificaciones podrían ser agregadas al diccionario, sin embargo la idea es que se registren los términos de manera breve pero concisa. En la figura 3.6 se muestra una plantilla para este documento.

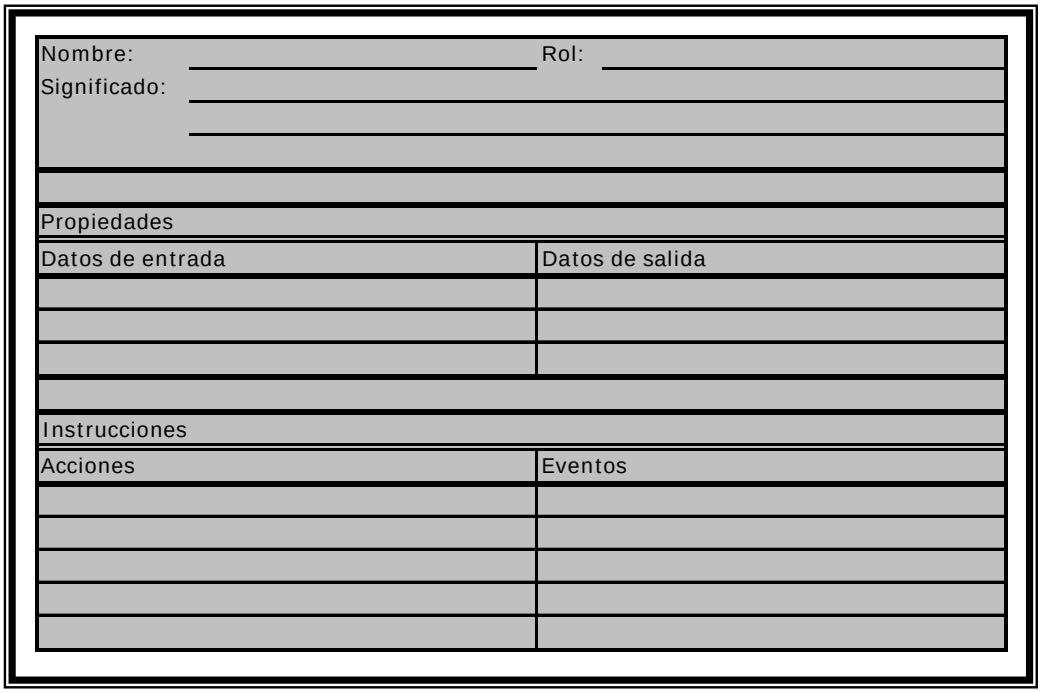


Figura 3.6. Plantilla para el diccionario del dominio

3.2.3. Arquitectura

La definición de la arquitectura debe responder a los requerimientos del Web Services y de alguna forma debe satisfacer los casos de uso. Dentro del contexto de Web Services, la arquitectura está orientada a los servicios. Tal como se mencionó en el capítulo 2, sección 2.1.5.

Recordando ese concepto, se identifican tres entidades: solicitadores del servicio, repositorio del registro y proveedores del servicio. Esta arquitectura puede ser representada como se muestra en la figura 3.7 donde se identifica cada entidad como un paquete estereotipado y se enlazan mediante relaciones estereotipadas, que dentro de éste contexto significa una relación de dependencia remota.

Al realizar este modelo lo relevante es que se identifiquen los servicios que serán publicados aunque en las primeras iteraciones estos pudieran no ser muy detallados (en etapas posteriores se irán refinando) y que se identifiquen las aplicaciones que serán clientes y los que serán los proveedores, es decir los solicitadores de los servicios y los proveedores de los servicios, en base a los requerimientos y los casos de uso definidos.

La arquitectura de un Web Service se puede representar mediante dos vistas: vista de pila y vista lógica (ver capítulo 2, sección 2.1.6 y 2.1.7)

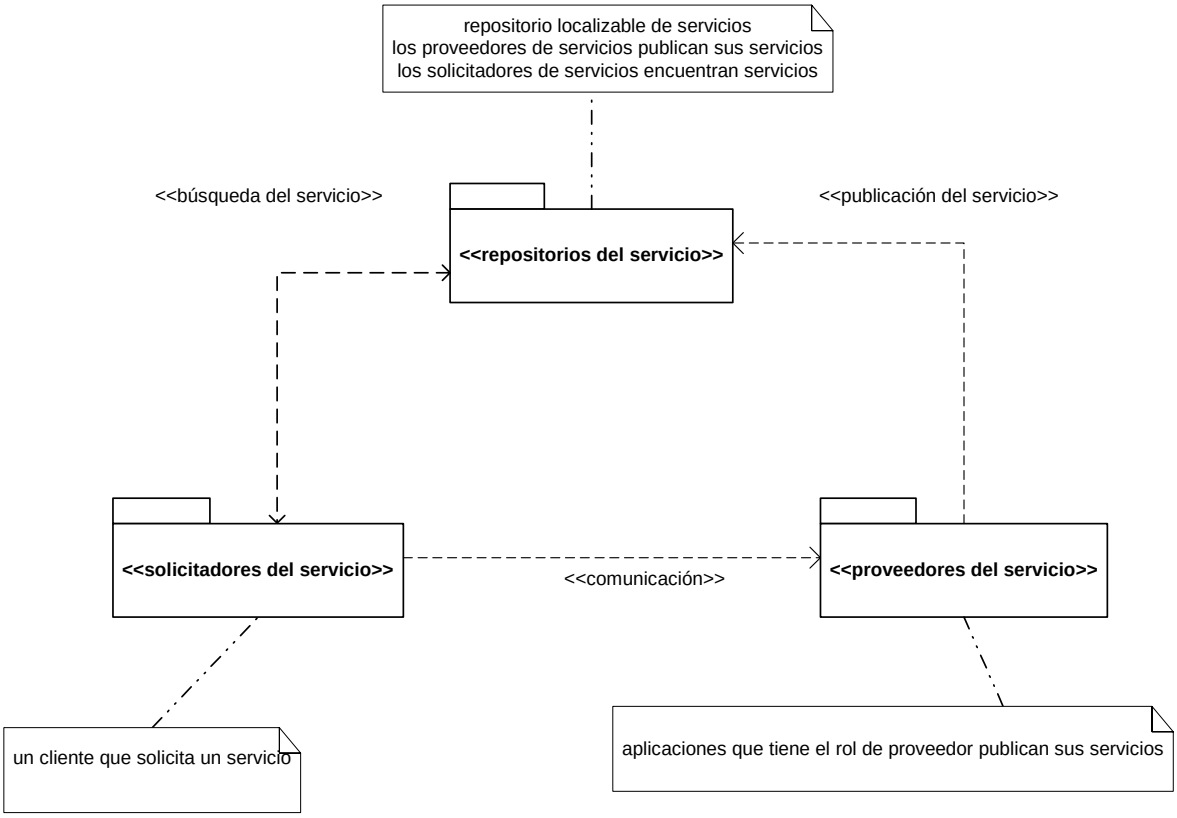


Figura 3.7. Modelación de la arquitectura orientada al servicio

Vista de pila

En esta vista se identifican seis capas apiladas una sobre otra, donde la comunicación e interacción se realiza de abajo hacia arriba. Es importante destacar el hecho de que la primera capa de abajo hacia arriba, está definida por la seguridad, la cual debe estar presente en el resto de las capas, es por eso que se ubica hasta abajo. Las siguientes capas a implementar de abajo hacia arriba son: transporte, envío de mensajes, descripción del servicio, descubrimiento (para el solicitador) ó registro (para el proveedor) y servicio.

Tanto el solicitador del servicio como el proveedor del servicio deben tener implementadas estas capas, de manera que la integración entre ellos resulte exitosa. En las figuras 3.8 y 3.9 se muestra ésta representación de capas para el proveedor del servicio y para el solicitador del servicio respectivamente. Dentro de estas figuras se explica lo que cada capa debe contener.

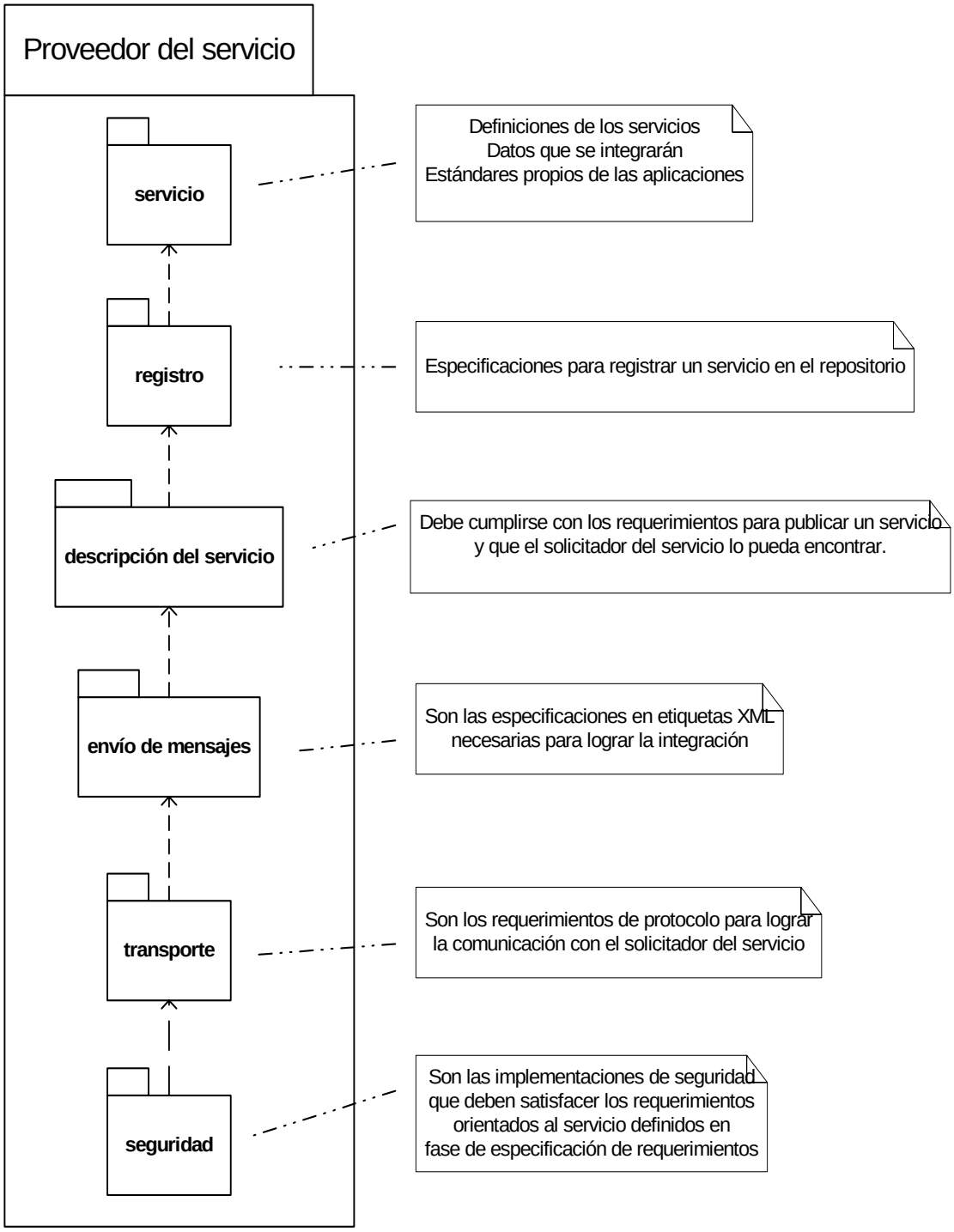


Figura 3.8. Modelación de la vista de pila para el proveedor del servicio

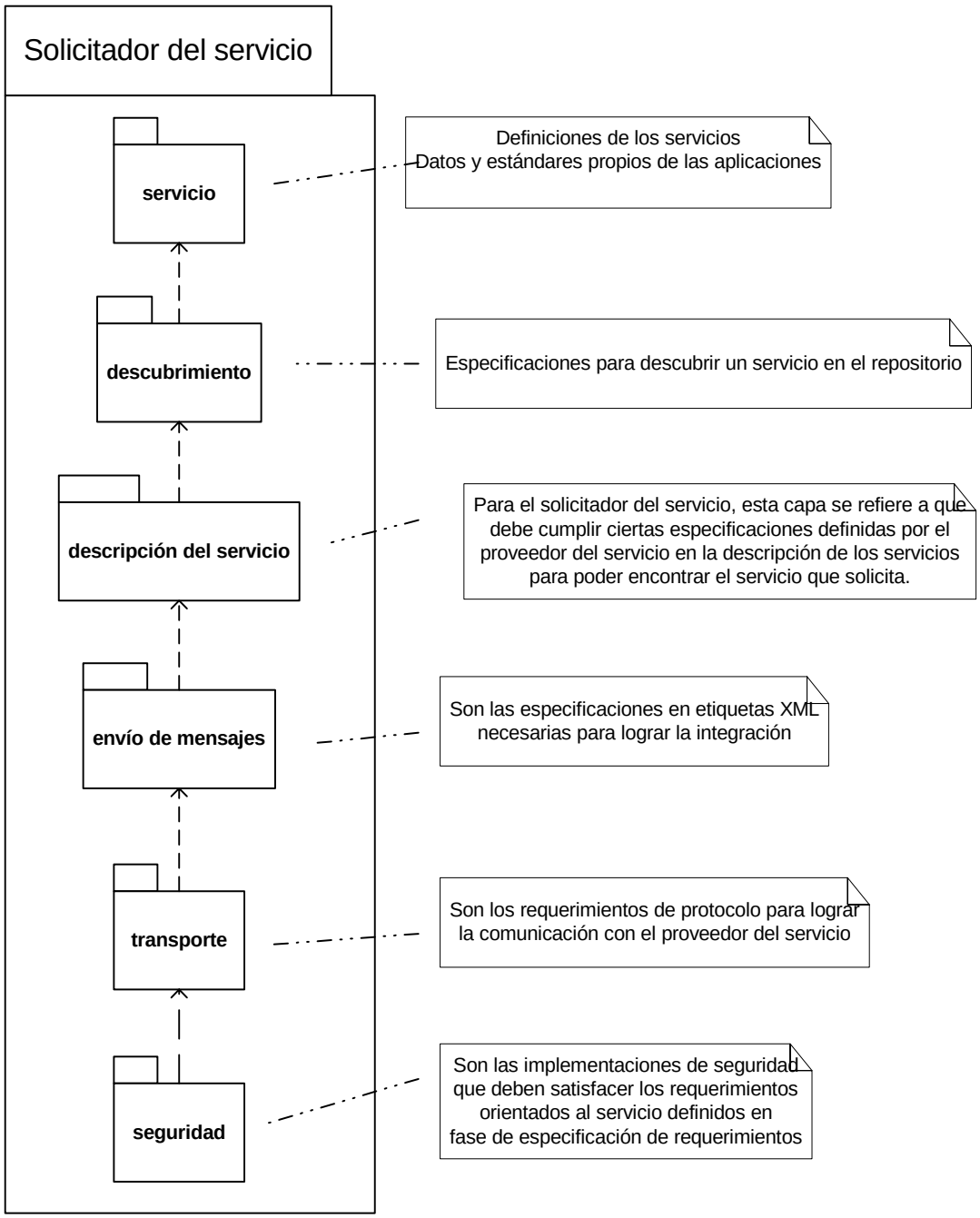


Figura 3.9. Modelación de la vista de pila para el solicitador del servicio

Cada una de estas capas tiene asociadas las tecnologías que forman el núcleo de Web Services, tal como se muestra en la figura 2.2. No obstante pudieran ser implementadas algunas otras.

En las primeras iteraciones es posible que no se tengan definidas claramente las relaciones entre las diferentes capas o que alguna de las capas pudiera no

estar definida completamente; sin embargo sí se pueden especificar aspectos fundamentales de estas capas y posteriormente se irán refinando los diagramas.

También es posible detallar más este tipo de capas, mediante clases que permitirán identificar aspectos específicos de cada capa. Esto se realiza mediante clases estereotipadas utilizadas en los diagramas de clases.

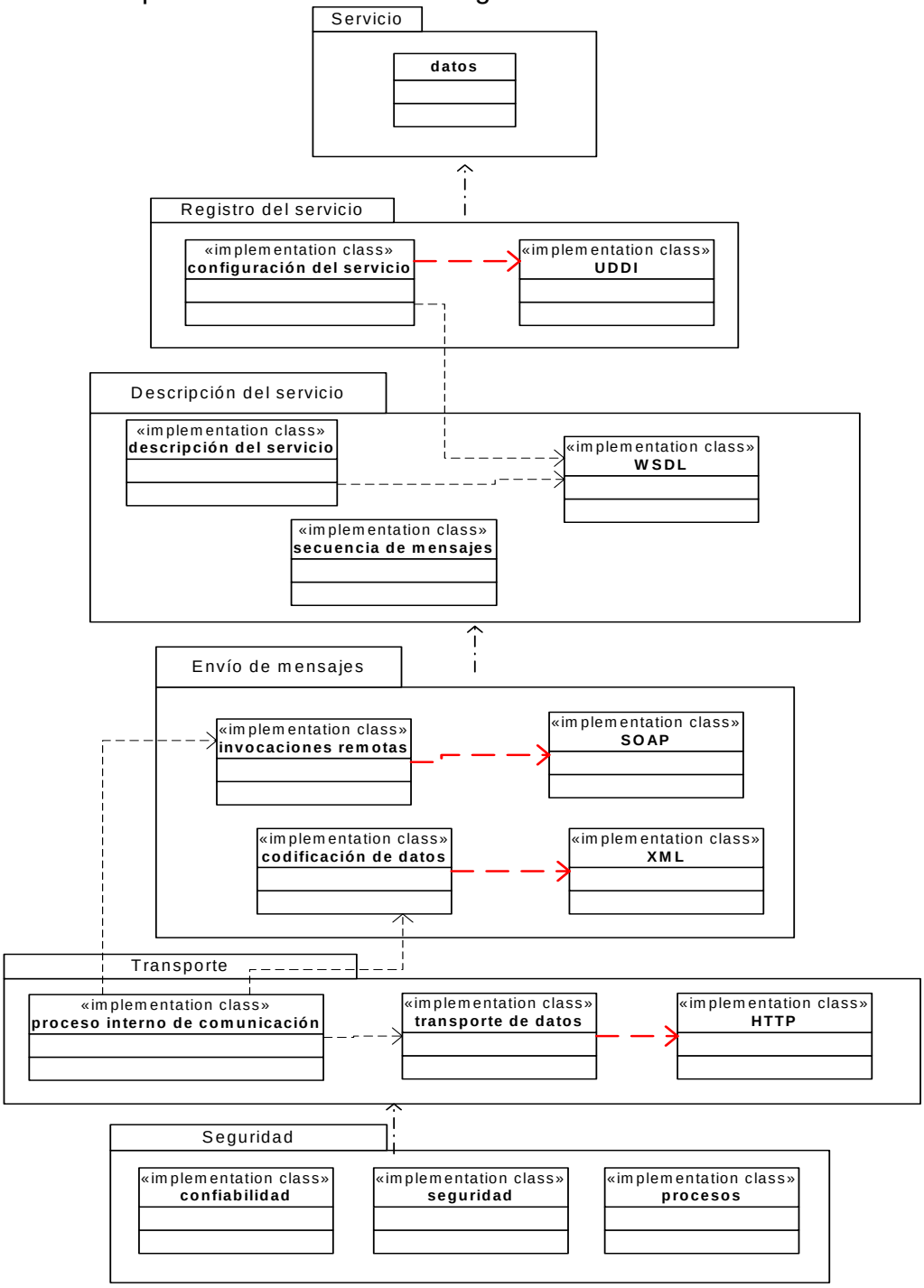


Figura 3.10. Modelación del diagrama de clases en la vista de pila para el proveedor del servicio

Respecto al solicitador del servicio, todas las capas excepto “registro del servicio”, podrán tener las mismas clases. En lugar de tener esta capa tiene “descubrimiento del servicio (figura 3.11).

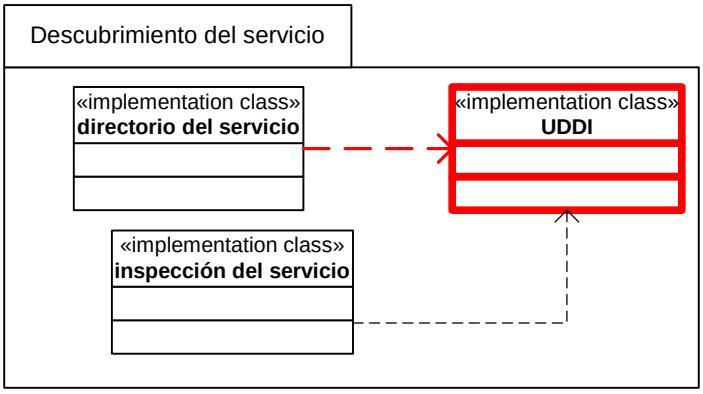


Figura 3.11. Modelación del descubrimiento de servicio en el diagrama de clases (vista de pila para el solicitador del servicio

Se utiliza clases estereotipadas en todas las capas, excepto para la representación de los servicios, debido a que no son clases ordinarias, sino que especifican que deberán ser implementadas conforme a las especificaciones de los estándares.

Estas clases podrán ser omitidas si el desarrollo lo requiriera o bien podrían ser agregadas algunas otras. Además recordar que las relaciones establecidas pueden no ser identificadas en la primera iteración.

Vista lógica

Desde esta perspectiva el objetivo de la definición de la arquitectura es definir el comportamiento de la interacción entre los Web Services para responder a las solicitudes de los servicios. (En el futuro los términos “solicitud” y “petición” se utilizarán indistintamente para denotar el mismo concepto):

a) Peticiones encadenadas. En este tipo de interacción los Web Services se comunicarán de manera lineal, es decir, una petición desencadenará la petición de otro servicio y ese de otro servicio hasta que se satisfaga la solicitud inicial. La figura 3.12 muestra este tipo de peticiones, se observa que un mismo Web Service puede hacer la función de proveedor si esta satisfaciendo una petición o cliente cuando esta solicitando un servicio (WS Proveedor/Cliente), en esta figura para que WS Cliente pueda satisfacer su

petición deberá realizarse primero la solicitud que hace el WS Proveedor/Cliente al WS Proveedor.

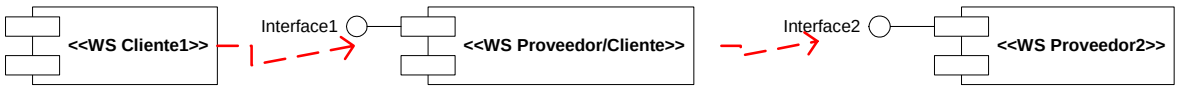


Figura 3.12. Vista lógica – peticiones encadenadas

b) Una petición es satisfecha por la unión de varios servicios. Una solicitud puede estar formada por la alianza de varios Web Services donde cada uno proporciona un servicio; sin embargo esta unión es transparente y la petición es satisfecha como si se tratara de un solo servicio proporcionado por un Web Service. La figura 3.13 muestra un servicio formado por varios e implementado en una misma interface para satisfacer la petición de WS Cliente 1.

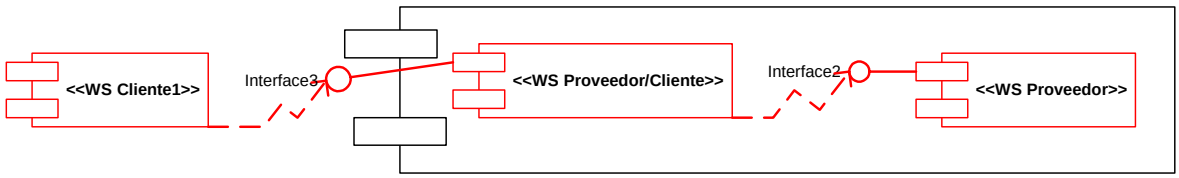


Figura 3.13. Vista lógica – petición satisfecha por un servicio formando por la unión de otros servicios

c) Peticiones en forma de árbol. Un Web Service tiene relación con uno o más Web Services que a su vez pueden tener alianzas con uno o más Web Services, derivando en peticiones con forma de árboles tal como se muestra en la figura 3.14, donde se observa que WS Cliente 1 dependerá de varios servicios que son del tipo lineal, varios servicios implementados como uno solo o bien en forma de árbol.

Esta clasificación está basada en la definida por Oellerman (ver sección 2.1.5.1). Se considera la funcionalidad del Web Service que se está desarrollando, sin tomar en cuenta la ubicación física real de los servicios.

Errores en la definición de la arquitectura

- Un error común es querer modelar las especificaciones de las tecnologías y no el sistema propiamente.

- Frecuentemente se define la arquitectura de un Web Service como si fuera una aplicación Web.

Estos errores fueron identificados durante la etapa de validación de la propuesta.

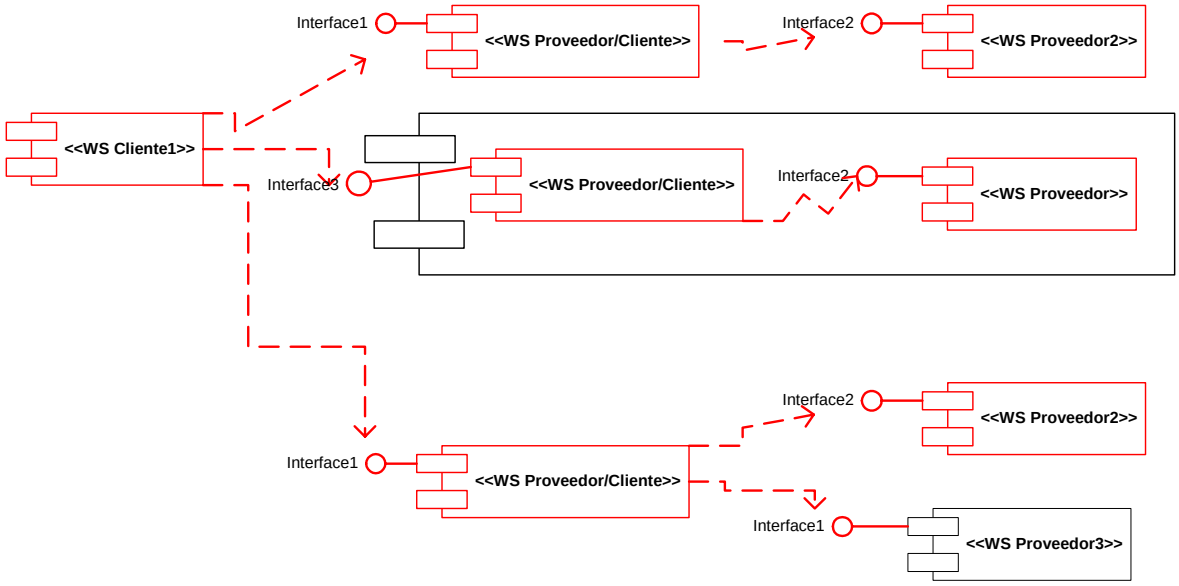


Figura 3.14. Vista lógica – peticiones en forma de árbol

3.2.4. Análisis

El objetivo de la fase de análisis es examinar los requerimientos de la aplicación, detallar los casos de uso para satisfacerlos y establecer un enlace hacia el diseño. El análisis de una aplicación distribuida se enfoca en los requerimientos funcionales del sistema, al igual que cualquier aplicación de otro tipo [Con1999].

3.2.4.1. Diagramas de clases

En está fase el diagrama de clases refinará los requerimientos especificados en los casos de uso, identificando las clases, sus atributos y sus relaciones.

Para la identificación de las clases se utilizan los casos de uso, el análisis de robustez y las clases que se haya definido en la fase de la arquitectura. Estos

diagramas podrán ser redefinidos en las diferentes iteraciones, se debe de tratar de agregar nuevas clases en vez de modificarlos.

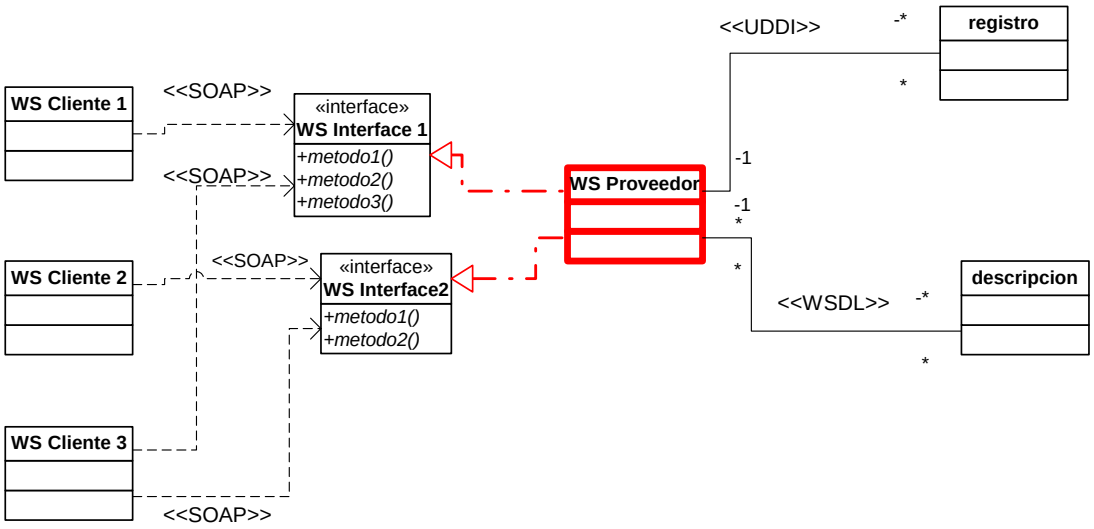


Figura 3.15. Diagrama de Clases

En la figura 3.15 se muestra un diagrama de clases en el que se tienen tres Web Services clientes que dependen de un Web Service proveedor, por medio de dos interfaces. Además por medio de relaciones estereotipadas está asociado con dos clases una llamada registro y otra llamada descripción. Las relaciones estereotipadas especifican el estándar mediante el cual se comunican.

En esta fase se pueden tener identificadas las interfaces con sus métodos, no obstante en iteraciones posteriores pueden ser modificados.

3.2.4.2. Diagramas de secuencia

Los diagramas de secuencia representan el primer enlace hacia el diseño, ya que el resultado define cómo el sistema deberá funcionar. Se basan en el análisis de robustez realizado y debe existir un diagrama de secuencia por cada caso de uso. Es por eso que no es recomendable empezar con estos diagramas sino se ha finalizado por completo el análisis de robustez para la iteración en particular.

Estos diagramas resultan adecuados para representar el envío de mensajes que implican el describir, registrar y descubrir servicios. Una forma de realizarlo, sugerida en [RS1999] es partir de la descripción de casos de uso, agregar los objetos interfaces y las entidades identificadas por medio de su respectivo análisis

de robustez y analizar en cada controlador el flujo de eventos y mensajes, los cuales definirán el funcionamiento del sistema.

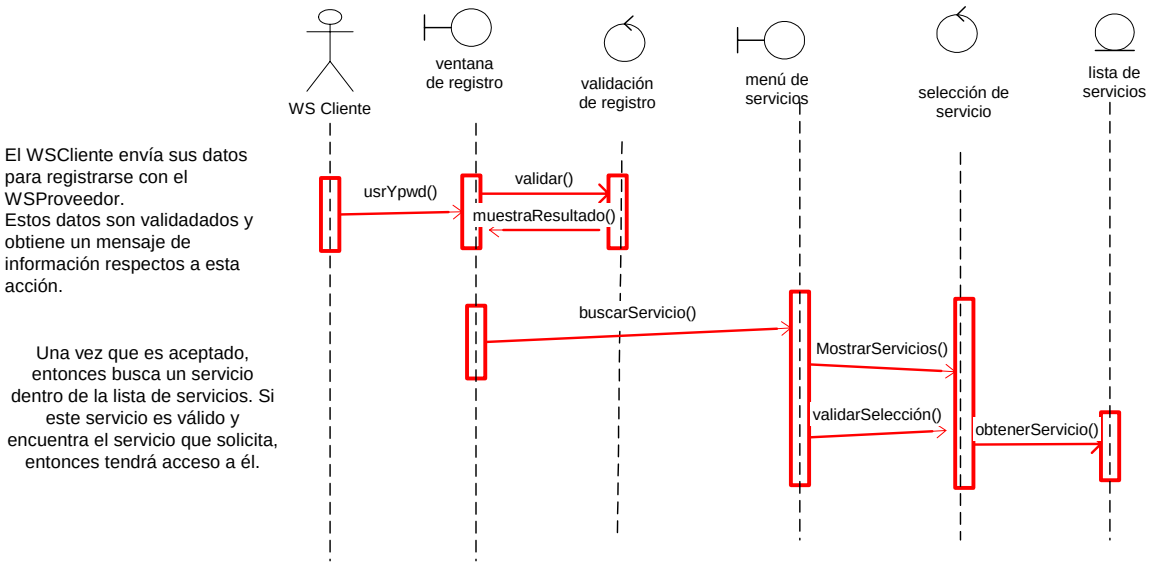


Figura 3.16. Diagrama de secuencia para el caso de uso “búsqueda de servicio”

La figura 3.16, muestra el diagrama de secuencia para el caso de uso definido en la sección 3.2.2.5 de este mismo capítulo. En esta figura se muestra que para realizar la búsqueda del servicio, ésta debe ser iniciada por el WS Cliente a través de identificarse con el WS Proveedor, una vez verificado el registro podrá buscar el servicio de su elección dentro de una lista de servicios disponibles. Se observa que algunos de los objetos pudieran ser omitidos, tal es el caso de la colección de servicios que forman parte del menú de servicios, la razón es que estos servicios van implícitos en la lista de servicios, por lo que pueden ser considerados en esa entidad.

3.2.4.3. Diagramas de actividad

Los diagramas de actividad son modelos dinámicos que muestran el flujo de una actividad a otra, en el desarrollo de Web Services estos diagramas resultan útiles para modelar los procesos locales y los procesos a los cuales podrán accesar otros Web Services. Esos procesos son los que fueron identificados en los casos de uso.

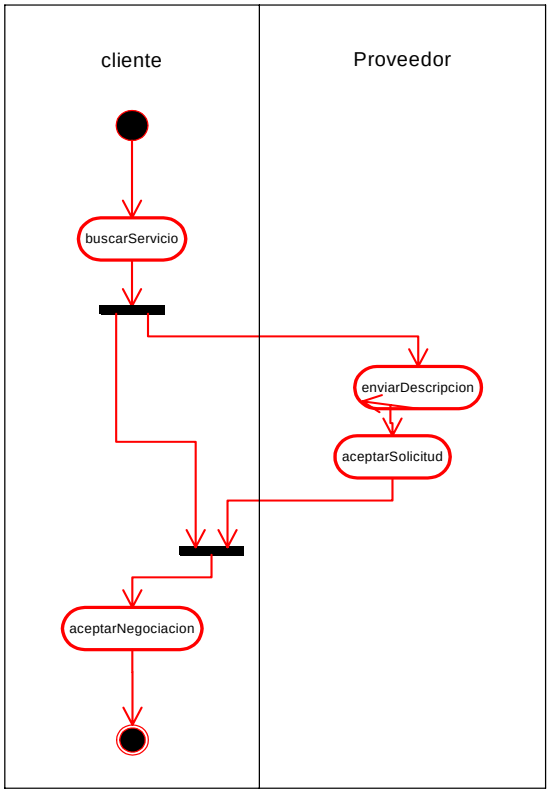


Figura 3.17. Diagrama de actividad para el caso de uso “búsqueda de servicio”

La figura 3.17 muestra el flujo del caso de uso “búsqueda de servicio”, desde la perspectiva del proceso. Se identifica que el cliente inicia el proceso por medio de la operación buscarServicio, esa operación requiere que el proveedor satisfaga la solicitud para lo cual hace uso de la descripción del servicio, si éste se encuentra disponible entonces le envía la respuesta al cliente, quien se encargará de aceptar la respuesta, en la figura lo hacen por medio de aceptaNegociación.

Estos diagramas representan el último enlace hacia el diseño, ya que modelan los eventos dentro de los procesos. En otras palabras los diagramas de actividad se derivan de los casos de uso y representan los requerimientos que deben ser satisfechos, proporcionando el cómo el sistema debe satisfacerlos.

Recomendaciones para el análisis

- Realizar un diagrama de secuencia por cada caso de uso [RS1999].
- Asociar la descripción del caso de uso a los diagramas de secuencia [RS1999].
- Sí modelar los diagramas de secuencia resulta complicado probablemente no existe aun buen análisis de robustez [RS1999].

3.2.5. Diseño

El diseño de Web Services deberá involucrar la arquitectura, lo que define la diferencia entre los artefactos obtenidos del análisis y del diseño. El diseño es donde la abstracción del negocio toma su primer paso hacia la aplicación real [Con2000]. Las dos entradas principales para esta etapa son los artefactos del análisis y los de la arquitectura.

3.2.5.1. Diagramas de clases

En el diseño los diagramas de clases son utilizados para identificar las operaciones y las relaciones existentes.

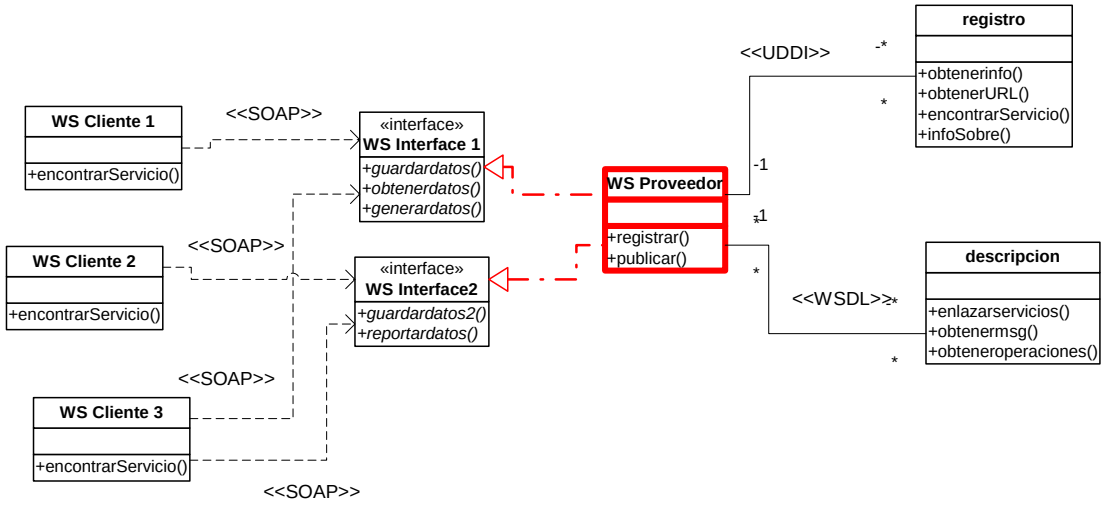


Figura 3.18. Diagrama de clases elaborado durante la fase del diseño

La figura 3.18 muestra un diagrama de clases que fue definido en la fase del análisis, donde ya se habían definido las interfaces a implementar, las relaciones existentes (recordemos la utilización de relaciones estereotipadas para especificar el estándar que se implementará) y las clases que contendrán los datos de los registros y de las descripciones. Sin embargo se observa que en esta fase se definen los métodos que cada clase deberá implementar. En el ejemplo las clases no tienen atributos, sin embargo desde la fase del análisis estos deben ser definidos.

Básicamente el objetivo de los diagramas de clase en esta fase es especificar todas las clases involucradas en el sistema, definiendo sus atributos, métodos y relaciones existentes entre ellos.

3.2.5.2. Diagramas de colaboración

Los diagramas de colaboración tienen la misma funcionalidad que los diagramas de secuencia, la diferencia es que estos enfatizan los objetos que participan en la interacción, mientras que los diagramas de secuencia enfatizan el tiempo.

Para los Web Services, sugerimos estos diagramas para modelar los mensajes que se enviarán entre los objetos. Estos objetos serán los identificados en el análisis de robustez, lo cual implica que en estos diagramas se utilizarán los objetos interface, entidades y controles ya definidos en fases anteriores, por lo que cualquier descubrimiento nuevo de alguno de esos elementos implicará modificar el análisis de robustez.

Una técnica sugerida por Larman en [Lar1999] para desarrollar este tipo de diagramas es la siguiente: elaborar un diagrama de colaboración para cada operación del sistema. Si el diagrama está resultando muy complejo es conveniente dividirlo en diagramas más pequeños.

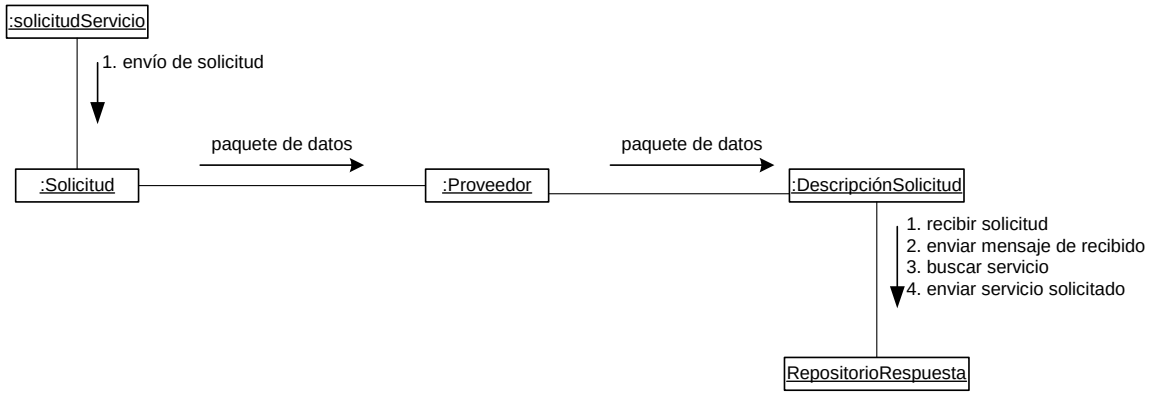


Figura 3.19. Diagrama de colaboración de la operación “utilización de un servicio”

En la figura 3.19 se muestra un diagrama de colaboración para la utilización de un servicio, donde se observa que :solicitudServicio inicia la operación enviando la solicitud al objeto :solicitud que será el encargado de enviarlo al :proveedor mediante paquetes de datos, éste último recurrirá a :DescripciónSolicitud para obtener la descripción y buscar el servicio dentro de :RepositorioRespuesta.

Es importante que los nombres utilizados correspondan con los identificados en los diagramas anteriores.

Recomendaciones para el diseño

- Verificar que al menos exista un caso de uso para cada requerimiento [RS1999].
- Realizar diagramas de secuencia por cada caso de uso [Lar1999].
- Realizar diagramas de colaboración para cada operación del Sistema [Lar1999].

3.2.6. Codificación e Implementación

En esta etapa se considera no solamente la escritura del código, sino también las tecnologías que involucrarán la implementación de la aplicación.

Es relevante aclarar que la codificación dependerá de cada sistema y estará en relación a la funcionalidad deseada. No obstante se sugiere en esta etapa utilizar los siguientes diagramas: los diagramas de componentes para dividir en módulos el sistema, los diagramas de clases identificados en el modelo del diseño, los diagramas de interacción y los diagramas de arquitectura para modularizar el sistema.

Los diagramas de componentes serán utilizados para agrupar en módulos al sistema tal como lo sugiere [RS1999] lo cual permitirá dividir el desarrollo entre el equipo y por otro lado los diagramas de clases se utilizarán para identificar las operaciones que cada clase debe realizar, al igual que las relaciones con otras clases.

Una forma muy común de implementar Web Services es utilizando portales y *portlets*. Un portal es usado generalmente para proporcionar acceso a los usuarios finales de la información, típicamente un portal despliega información originaria de varias fuentes. Y dependiendo de la configuración del usuario se pueden utilizar *portlets* (áreas rectangulares que despliegan información), los cuales se incluyen en el portal.

Utilizar esta implementación permite implementar Web Services de dos formas diferentes: los *portlets* son utilizados como interface final hacia el usuario de los Web Services o bien, los *portlets* son descritos, publicados y registrados como Web Services. Un ejemplo del primer caso es cuando el *portlet* se ejecuta de manera local sobre el portal y usa un Web Service para accesar a la información. Para el segundo caso un ejemplo es cuando un *portlet* remoto está disponible para los usuarios del portal sin requerir de una instalación local del *portlet*.

3.2.7. Pruebas

Es la etapa final del proceso y probablemente la más difícil, ya que el objetivo es que el producto final satisfaga los requerimientos.

Aún cuando la selección de las pruebas dependerá del equipo de desarrollo, algunas de las pruebas que se podrían aplicar son las sugeridas por los diferentes autores descritas en el capítulo 2. Es decir, aplicar pruebas de unidad, de caja blanca (diagramas de casos de uso), de caja negra (diagramas de casos de uso), pruebas del sistema, pruebas de integración (diagramas de componentes, diagramas de la arquitectura) y pruebas de aceptación, entre otros. La definición de estas pruebas se encuentra en la sección 2.3.2 del capítulo 2.

3.3. Resumen

- Un proceso es una generalización de una serie de pasos específicos, conocida como metodología.
- En este capítulo se presenta una propuesta de desarrollo de Web Services que responde a las preguntas qué hacer, cómo hacer y cuándo hacer.
- Un artefacto son los diagramas y documentos entregables, resultados de cada fase.
- Existen dos documentos que no forman parte del proceso pero que son importantes desarrollar, estos son: introducción y panorama general del sistema.
- Las fases y artefactos del proceso propuesto son presentados en la siguiente tabla:

Fase	Artefactos
Requerimientos	ï clasificación de requerimientos ï identificación de los actores y los casos de uso ï diagramas de casos de uso ï descripción de los casos de uso ï análisis de los casos de uso ï diccionario del dominio.
Arquitectura	ï modelado de la arquitectura orientada al servicio (diagramas de paquetes) ï vista de pila (diagramas de paquetes y diagramas de clases) ï vista lógica (diagramas de servicios).
Análisis	ï diagramas de clases ï diagramas de secuencia ï diagramas de actividad.
Diseño	ï diagramas de clases ï diagramas de colaboración.
Codificación	ï Diagramas de componentes ï Diagramas de clases (definidos en el diseño).
Implementación y Pruebas	ï Depende del equipo de desarrollo

Tabla 2. Fases del proceso propuesto

En el siguiente capítulo se presenta la validación a la propuesta aquí desarrollada. La cual se realiza mediante dos casos de estudio presentados en el siguiente capítulo.

Capítulo 4 Validación de la propuesta utilizando Casos de Estudio

En este capítulo se muestran dos casos de estudio, mediante los cuales se valida el proceso propuesto. Uno de los casos de estudio (Consorcio MedBiquitous) es extraído de un sistema ya implementado y el otro caso de estudio (Integración de contenidos curriculares distribuidos) es un sistema en desarrollo.

En ninguno de los dos casos se llegó a la fase de la codificación e implementación y como consecuencia tampoco a la fase de pruebas, no obstante el proceso propuesto se valida mediante la modelación de las otras fases.

A continuación se describen los casos de estudio, los principales casos de usos y sus diagramas correspondientes. La documentación completa de primer caso de estudio se encuentran en el Anexo C “Consorcio MedBiquitous”.

4.1. Caso de estudio:
“Consorcio MebBiquitous”

4.1.1. Introducción

Este caso de estudio fue obtenido de una sistema ya implementado, sin embargo en este trabajo se valida mediante el proceso propuesto en el capítulo anterior.

Situación

El Consorcio MedBiquitous es una organización no lucrativa fundada en mayo del 2001, creado por la Escuela de Medicina Johns Hopkins en asociación con otras sociedades médicas. Su misión es desarrollar tecnología para las comunidades relacionadas con la medicina que aporten educación profesional, experiencias laborales y, que colaboren para mejorar el nivel de salud en los pacientes. Para cumplir esta misión MedBiquitous ofrece una comunidad de comunidades, es decir, un portal integrado por comunidades médicas, sociedades que pueden ser formadas por médicos y/o por universidades y, entidades gubernamentales e industriales.

Solución

El portal desarrollado por MedBiquitous permite a los miembros trabajar juntos e intercambiar información de forma fiable, segura y automática. Este es una plataforma consistente de varias tecnologías integradas: XML permite a las organizaciones involucradas mejorar el intercambio de datos, un framework en Java que permite incrementar la flexibilidad del software y mejorar el desarrollo, y Web Services que establece un canal de intercambio de información y de servicios disponibles por diferentes miembros a través de sus portales.

Beneficios

- La comunidad de comunidades desarrollada permite compartir información relevante de las últimas investigaciones realizadas por diferentes doctores que no están físicamente en el mismo lugar.

- El portal implementado facilita la capacitación continua, mediante los cursos en líneas disponibles y las publicaciones de *journals*.

- Utilizando este medio no lucrativo, los miembros pueden incrementar sus conocimientos y compartir el resultado de sus avances.

- Este portal define grupos de discusión disponibles en línea para todos los miembros.

- Un miembro puede participar en votaciones sobre políticas y estándares técnicos de Medicina.

4.1.2. Panorama general

Objetivo

Compartir resultados de investigaciones, *journals* y cursos en línea y participar en grupos de discusión mediante un portal confiable e independiente de plataformas de desarrollo.

Nombre

WSMB: Web Services MedBiquitous

Descripción de la aplicación

Mediante el desarrollo de Web Services se integrarán las aplicaciones de diferentes socios (comunidades, sociedades y entidades) de MedBiquitous. A través del Web Service (que para futuras referencias indistintamente será llamado portal), los miembros registrados podrán obtener los beneficios especificados en la sección anterior.

En cuanto a los estándares necesarios se han identificado la utilización de XML, Framework en Java y los estándares tecnológicos propios de Web Services.

Identificación de entidades y recursos

Se identifican tres entidades principales: WSMB recibirá información desde
1) sociedades (de profesionistas médicos o de universidades médicas),
2) organizaciones gubernamentales y/o industriales (p.e. hospitales y laboratorios)
3) otras comunidades médicas.

Los recursos que se desearán compartir son:
1) *journals*
2) cursos en línea
3) grupos de discusión
4) resultados de investigaciones

4.1.3. Requerimientos

a) Tecnológicos

SOAP
La versión utilizada es SOAP 1.1.

Mediante este estándar se permitirá que las aplicaciones puedan transmitir sus datos por medio de sobres, que especificarán las condiciones de solicitud y respuesta a solicitudes de servicios.

UDDI
La versión utilizada es UDDI V3.0.

Para WSMB los servicios deberán ser publicados en forma privada, ya que solo las aplicaciones que estén registradas como miembros podrán tener acceso a los servicios ofrecidos.

WSDL
La versión utilizada es WSDL 1.1.

Por medio de WSDL se detallaran las descripciones de los servicios, que contendrán las especificaciones de la implementación y de las interfaces de los servicios.

a) De integración

A continuación se especifican los servicios aportados por cada entidad:

- Una comunidad podrá publicar *journals* y resultados de investigaciones, ofrecer cursos en línea, formar y administrar grupos de discusión.

- Una sociedad podrá publicar *journals* y resultados de investigaciones y, ofrecer cursos en línea.

- Una organización gubernamental/industrial podrá publicar resultados de investigaciones y ofrecer cursos en línea.

WSMB recibirá solicitudes a servicios desde diferentes sociedades, comunidades u organizaciones industriales y gubernamentales, aún cuando algunas solicitudes pudieran coincidir estas serán específicas por lo que requerirán también de respuestas específicas. La figura 4.1 muestra las interfaces que estarán implementadas en WSMB y la dependencia que existirá con respecto a cada entidad; en este diagrama se modelan los servicios que cada entidad aportará.

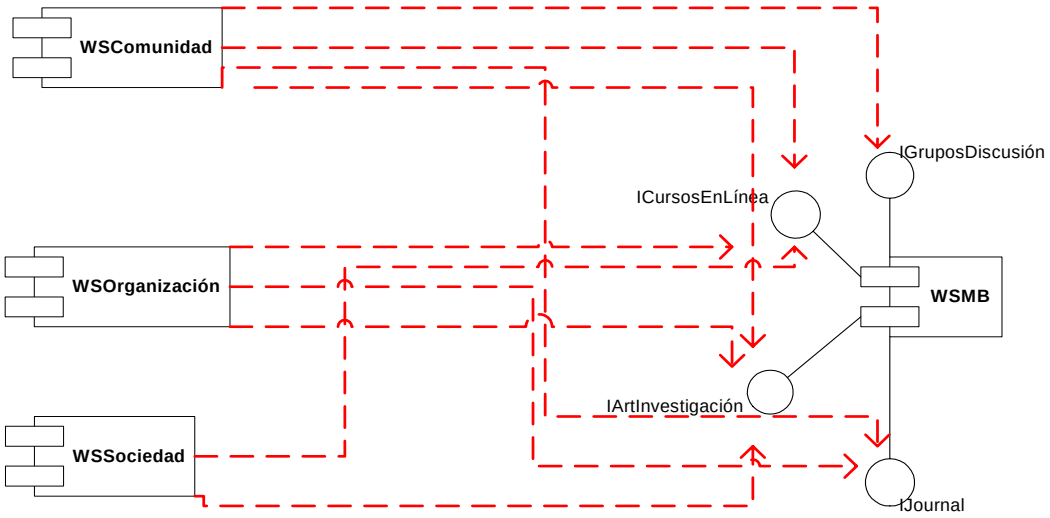


Figura 4.1. Diagrama de integración para WSMB

c) Orientados al servicio

Seguridad

Los miembros que accedan a WSMB, podrán tener la certeza de que su información no será descubierta por otras personas que pudieran hacer mal uso de sus datos, por otro lado, tendrán acceso privado a los servicios, para lo cual será necesario proveer nombre de usuario y clave a cada miembro, para las entidades cliente.

Por otro lado WSMB autenticará el origen de cada servicio, además de garantizar que la información enviada no podrá ser interceptada por intrusos. También verificará que las entidades quienes estén ofreciendo un servicio cuenten con los requerimientos necesarios para hacerlo.

Una tecnología que podría ser implementada es HTTPS, que combina HTTP Basic Authentication (HTTP-AUTH) con Secure Socket Layer (SSL). HTTP-AUTH autoriza el acceso mediante una clave codificada con Base64; sin embargo por sí sólo no es un método seguro por lo que es necesario combinarlo con SSL para hacer un método más fuerte en cuanto a seguridad se refiere. SSL satisface los requerimientos de confidencialidad, autenticación e integridad.

Confiabilidad

Se esperará que cualquier miembro pueda realmente acceder a los servicios definidos en secciones anteriores. Es decir evitar que se tengan ligas a journals, cursos en línea, grupos de discusión o publicaciones de resultados de investigaciones no disponibles.

4.1.3.1. Casos de uso

Identificación de los actores y casos de uso

En la definición de los casos de uso, la primera actividad es la identificación de los actores, que son todas aquellas entidades externas que interactuarán con el sistema.

Actores

- AppOrganización, representa a la aplicación correspondiente a las organizaciones.
- AppComunidad, representa a la aplicación correspondiente a las comunidades.
- AppSociedad, representa a la aplicación correspondiente a las sociedades.
- Administrador del portal, se encargará del registro de las aplicaciones.

Identificación de Casos de uso

Se agrupan los casos de uso de la manera siguiente:

a) Búsqueda de servicios. Cuando cualquier aplicación acceda al portal y requiera buscar uno de los recursos ofrecidos (figura 4.2):
   Cursos en línea
   Consultar grupos de discusión
   Consultar *journals*
   Consultar artículos de investigación

b) Publicación de servicios. Cuando cualquier aplicación desee poner a disposición del resto de la comunidad de comunidades algún recurso:
   Ofrecer cursos en línea
   Crear grupos de discusión
   Agregar *journals*

Agregar artículos de investigación

c) Uso de servicios. Cuando cualquier aplicación desee utilizar alguno de los recursos ofrecidos:
Participar en grupos de discusión
Obtener *journals* y artículos de investigación
Inscripción a cursos en línea

d) Registro de aplicaciones. Cuando una nueva aplicación quiera pertenecer a la comunidad de comunidades o bien, dejar de formar parte de ella:
Alta de aplicación
Baja de aplicación
Actualiza datos

4.1.3.2. Diagramas de casos de uso

La figura 4.2 muestra el diagrama para los casos de uso en los que los actores AppComunidad, AppSociedad, AppOrganización desean invocar las consultas de los cursos en línea, grupos de discusión, journals o artículos de investigación, estos casos de uso fueron clasificados como búsqueda de servicios. En la figura se observa que cualquier de los actores podrían realizar las invocaciones hacia la consulta de los servicios.

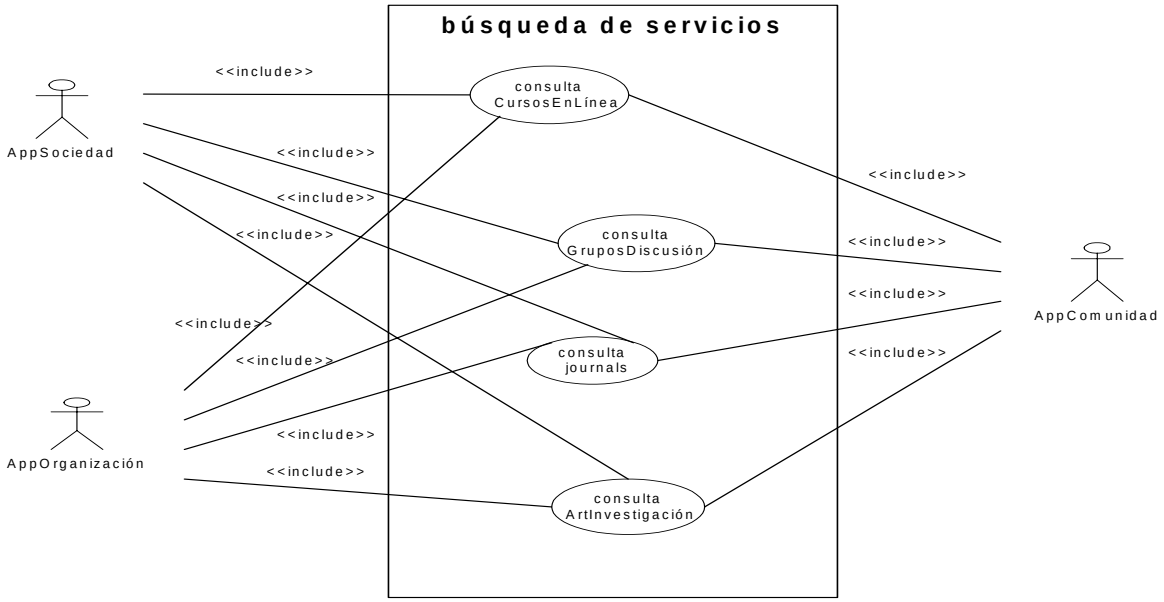


Figura 4.2. Diagrama de caso de uso “Búsqueda de servicios”

4.1.3.3. Descripción de los casos de uso

En la figura 4.3 se muestra la descripción del caso de uso “consulta grupos de discusión”, se observa que se utiliza la plantilla sugerida en el capítulo 3.

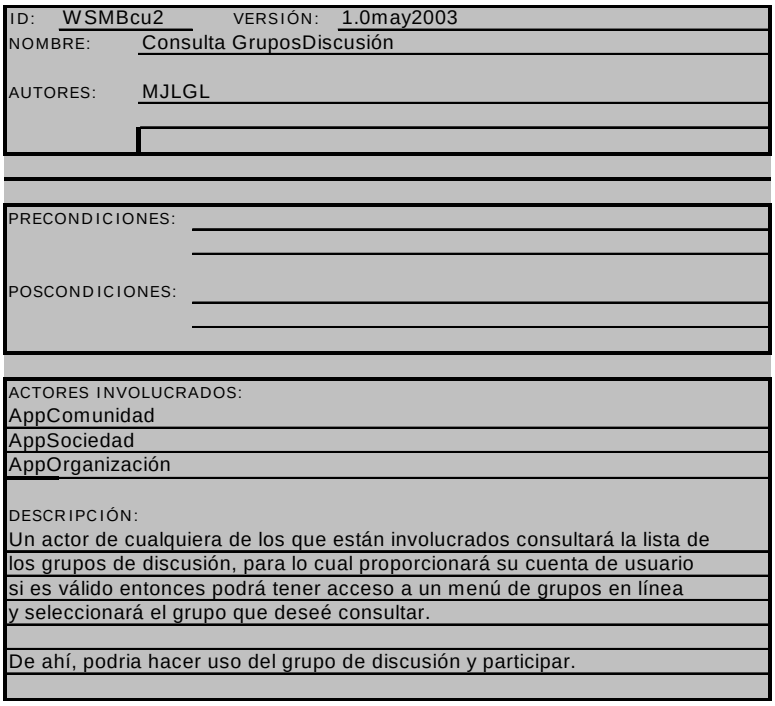


Figura 4.3 Descripción del caso de uso “consulta grupos de discusión”

4.1.3.4. Análisis de robustez

En la realización del análisis de robustez, se debe modelar un diagrama para cada caso de uso definido. La figura 4.4 muestra el análisis de robustez para la búsqueda de servicios, en este diagrama se observa que un caso de uso puede estar relacionado con otro. El objeto interface “cuadro de registro” inicia el curso de la acción, después de que haya sido validada la clave de acceso al portal se desplegará el “menú de servicios” representado por una entidad, la validación se representa con el controlador “validación de registro”. El actor podrá seleccionar el grupo de discusión que desee consultar y eventualmente podría utilizar el caso de uso “participar grupo de discusión”.

Búsqueda de servicios: Consulta grupos de discusión

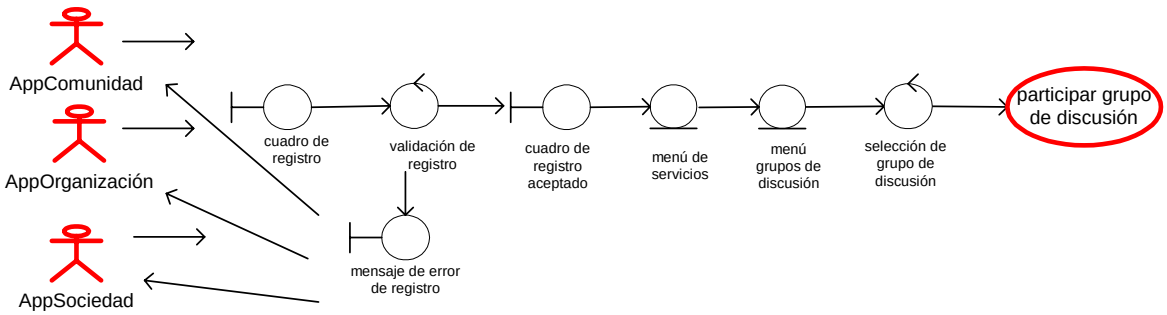


Figura 4.4. Diagrama de análisis de robustez “Consulta grupos de discusión”

4.1.3.5. Diccionario del dominio

El diccionario del dominio ha sido omitido por ser un documento técnico y el propósito principal de este trabajo es demostrar la validez del proceso en el sentido de la aplicabilidad de los diagramas; sin embargo esto no significa que no sea importante o pueda no realizarse.

4.1.4. Arquitectura

Vista de pila

Servidor

La figura 4.5 muestra la vista de pila para el proveedor del servicio, en la cual se identifican las seis capas de la arquitectura, cada capa tiene comentado los aspectos que deben ser considerados para su implementación.

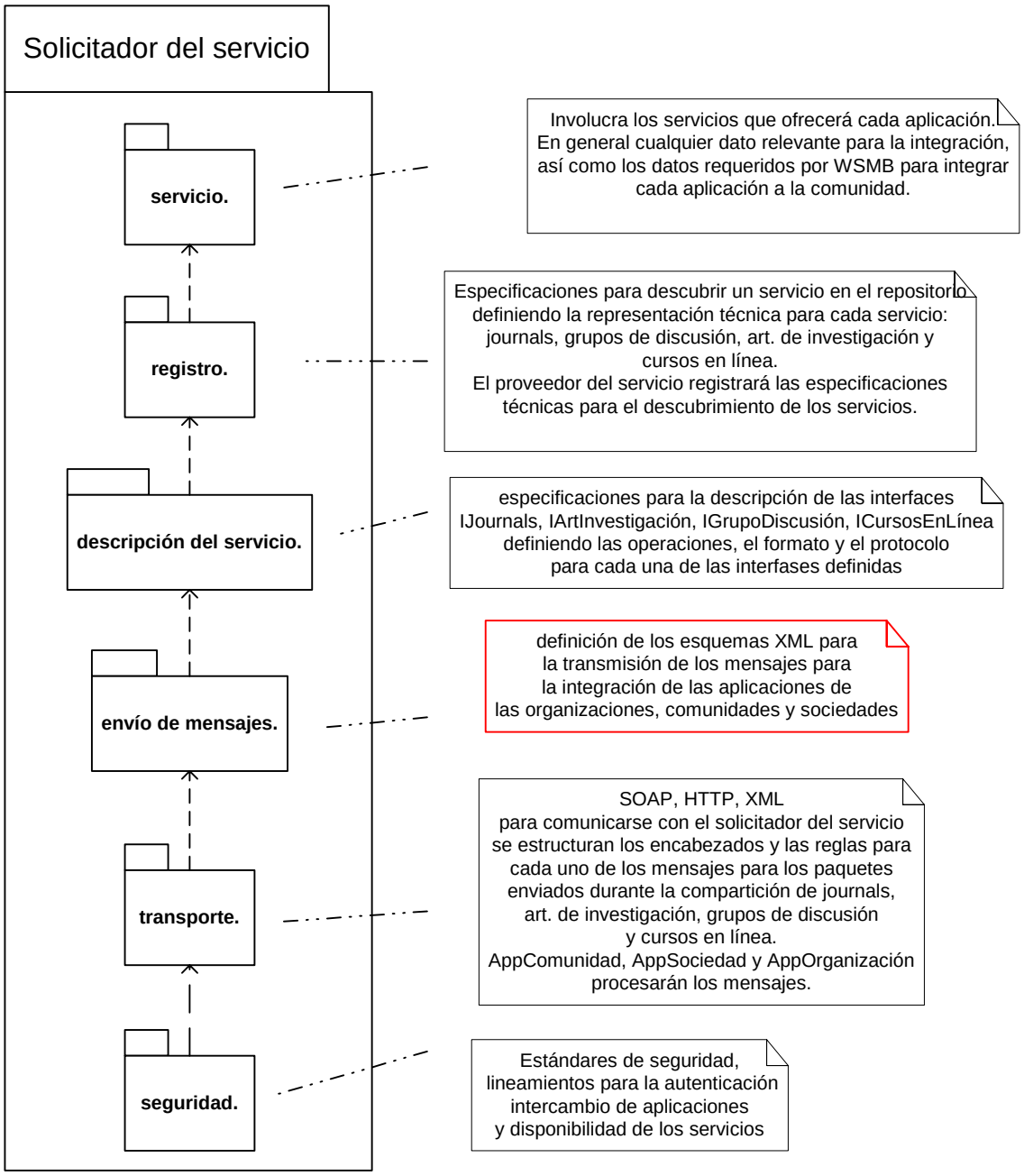


Figura 4.5. Diagrama de la vista de pila para el proveedor del servicio

Cliente

En la figura 4.6 muestra la vista de pila para el solicitador del servicio. Al igual que en la vista de pila para el proveedor, cada capa tiene comentados los aspectos que deben ser implementados.

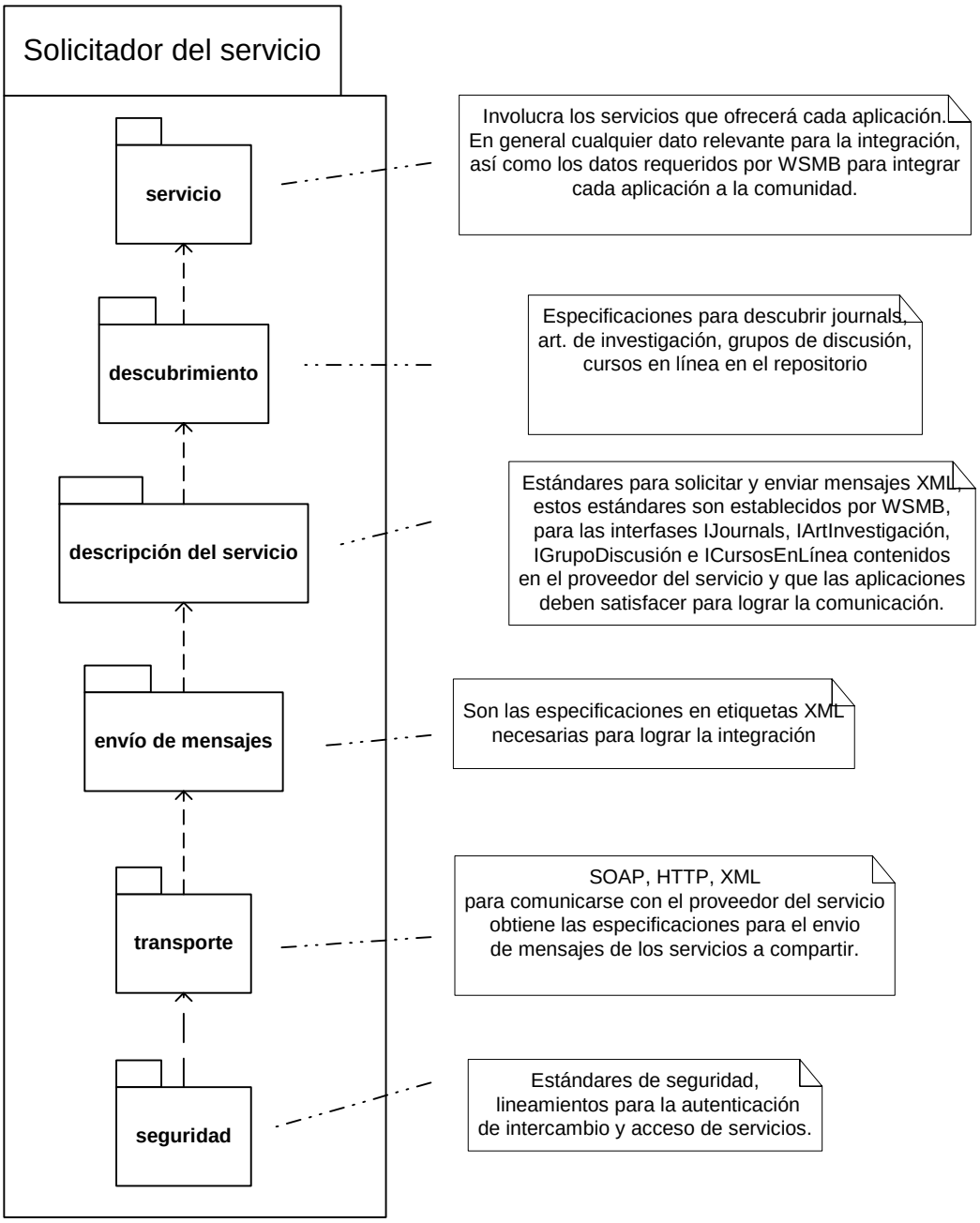


Figura 4.6. Diagrama de la vista de pila para el solicitador del servicio

Vista lógica

La figura 4.7 muestra la vista de servicios (vista lógica) para WSMB en la que un Web Services Cliente que utilice el portal y utilice los servicios lo hará como si se tratará de una sola interface, aunque el portal internamente tendrá invocaciones hacia otros servicios. Se observa que este diagrama es una modelación diferente para el diagrama realizado durante la especificación de los

requerimientos de integración. En otras palabras la modelación de los requerimientos de integración es útil para representar esta vista de la arquitectura.

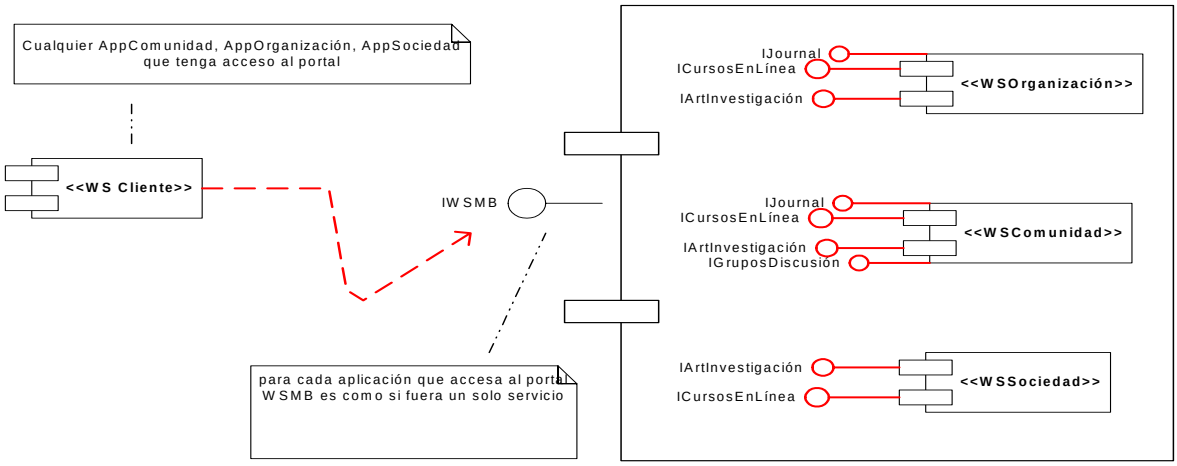


Figura 4.7. Diagrama de la vista lógica (vista de servicios)

4.1.5. Análisis

4.1.5.1. Diagramas de clases

En el diagrama de clases se definen las clases que implementarán y utilizarán las interfaces y se identifican clases correspondientes a los estándares WSDL y UDDI. Se observa que en el diagrama 4.8 no se han identificado atributos (solo la clase entidad tiene algunos atributos) y que a excepción de las relaciones estereotipadas para representar los estándares no se representan más relaciones entre clases. En esta figura se identifican la clase WSMB que realiza cuatro interfaces las cuales serán implementadas por las entidades, entre éstas clases se identifica la asociación <<SOAP>> la cual permite identificar el estandar que definirá la relación, se observan las clase registro relacionadas por la asociación estereotipada <<UDDI>> y la clase descripción que se relaciona por la asociación <<WSDL>>.

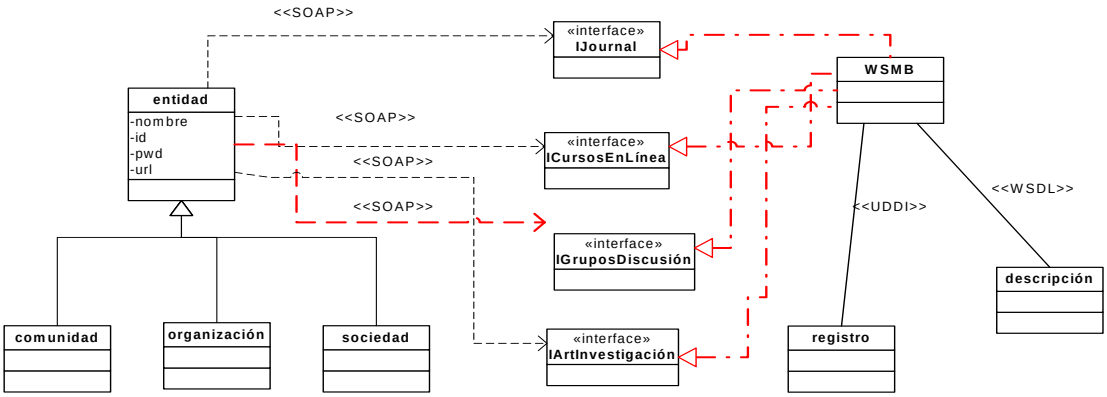


Figura 4.8. Diagrama de clases en la fase del análisis

4.1.5.2. Diagramas de secuencia

En esta sección se muestran el diagrama de secuencia para la consulta de los grupos de discusión. La figura 4.9 muestra el flujo de mensajes entre los objetos identificados en el análisis de robustez. El objetivo de este diagrama es mostrar la secuencia de mensajes que desencadenan una acción tal como se muestra en la figura.

Búsqueda de servicios: Consulta grupos de discusión

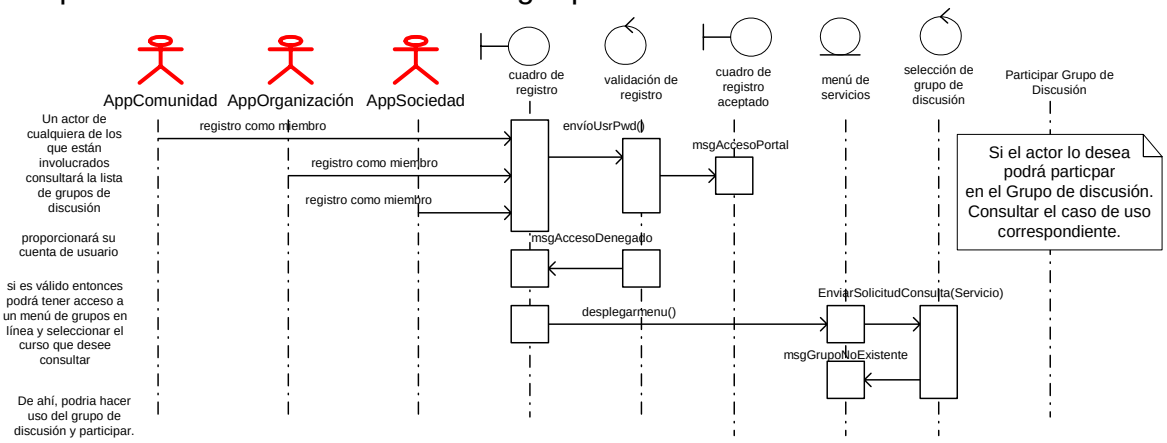


Figura 4.9. Diagrama de secuencia para “consulta grupos de discusión”

4.1.5.3 Diagramas de actividad

En la identificación de los casos de uso estos fueron divididos en cuatro grupos: búsqueda de servicios, publicación de servicios, uso de servicios y registro de aplicaciones. En la figura 4.10 se muestra el diagrama de actividad correspondiente a la búsqueda de servicios, en esta figura se identifican las actividades necesarias por parte de los clientes (AppComunidad, AppSociedad, AppOrganización) y por parte del proveedor WSMB para la realización de la búsqueda de servicios. Los clientes inician esta búsqueda y el WSMB a través de las interfaces deberá obtener las descripciones para poder satisfacer la solicitud del servicio y que el cliente pueda satisfacer su petición.

Búsqueda de servicios

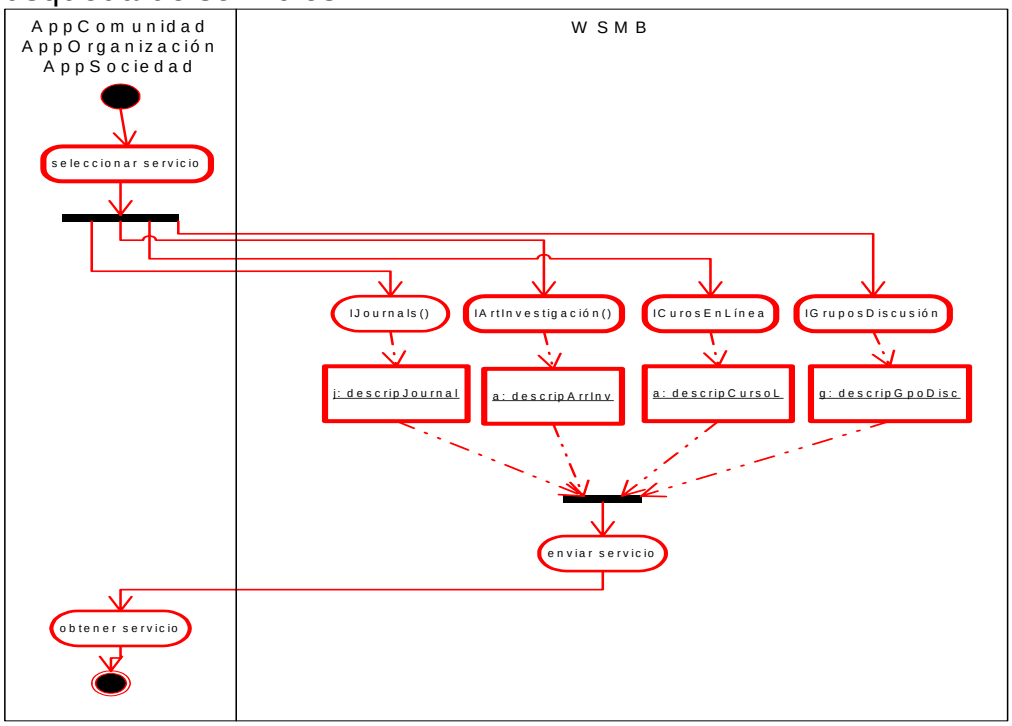


Figura 4.10. Diagrama de actividad para la búsqueda de servicios

4.1.6. Diseño

4.1.6.1. Diagramas de clases

La figura 4.11 muestra el diagrama de clases elaborado en la fase del análisis, pero en esta fase se definen las operaciones que contendrá cada interface que finalmente son los métodos a los que los clientes podrán accesar y mediante los cuales WSMB integrará las aplicaciones y permitirá la compartición de servicios. Se identifican también las operaciones de las clases registro y descripción.

Es importante aclarar que para fines ilustrativos en este diagrama se han omitido los atributos, sin embargo para esta fase ya deben estar definidos.

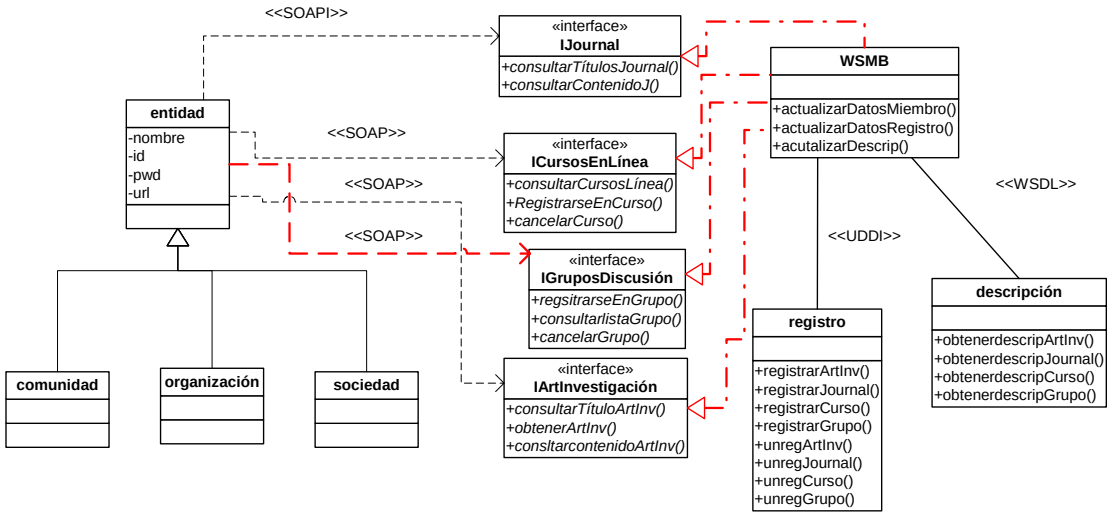


Figura 4.11. Diagrama de clases elaborado en la fase de diseño

4.1.6.2. Diagramas de colaboración

En la identificación de los casos de uso estos fueron divididos en cuatro grupos: búsqueda de servicios, publicación de servicios, uso de servicios y registro de aplicaciones. En la figura 4.12 se muestra el diagrama de colaboración correspondiente a la búsqueda de servicios. Inicialmente se solicita el servicio, se obtendrá la solicitud que podría corresponder a cualquiera de las interfaces que implementen los servicios, una vez que se tiene la solicitud del servicio buscado se obtiene la descripción que es enviada al repositorio para que sea satisfecha esta solicitud y regrese al objeto que inició la interacción.

Búsqueda de servicio

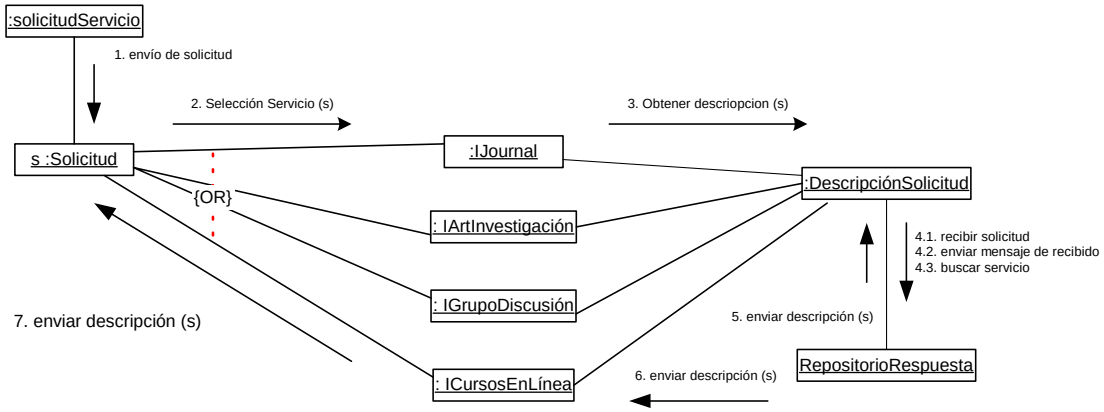


Figura 4.12. Diagrama de colaboración para la búsqueda de servicio

Los diagramas de colaboración están directamente relacionados con los diagramas de actividad elaborados en la fase del análisis, la relevancia de estos diagramas es que se identifican los eventos de un objeto a otro. Y estos permitirán definir las operaciones o métodos que deberán existir para la compartición de los servicios.

4.1.7. Codificación e Implementación y Pruebas

Estas fases han sido omitidas debido a que no se llegó a implementar físicamente dicho sistema. Aún cuando este sistema ya está implementado; en este trabajo se realizó la modelación con el proceso propuesto en el capítulo 3. Sin embargo algunos diagramas que podrían ser utilizados durante la codificación son los diagramas de integración, diagramas de arquitectura, diagramas de colaboración y los diagramas de clases.

4.2. Caso de estudio:
“Integración de contenidos curriculares distribuidos”

4.2.1. Introducción

Situación

- Falta de esquema de colaboración entre docentes
- Intercambio de materiales sin administración
- Duplicidad de esfuerzos en la construcción de los cursos

Solución

Implantación de un sistema descentralizado administrativa y operativamente que permita el intercambios de los contenidos de los cursos y del material de apoyo.

Beneficios

- Reutilización del material
- Actualización dinámica de los contenidos
- Establecimientos de una academia conformada académicamente
- Optimización de los recursos docentes
- Agilización en el proceso de construcción de contenidos

4.2.2. Panorama General

Objetivo

Compartir los contenidos y los materiales de apoyo de los cursos de diferentes universidades.

Nombre

WSU: Web Service Universidad

Descripción de la aplicación

Se utilizará Web Service para integrar cursos (material y contenidos) provenientes de diferentes universidades (sección Noroeste, sección Norte, sección Noreste, sección Centro, sección Occidente, sección Golfo, sección Mayab, sección Sur). Mediante tecnología Web Services esta información podrá estar disponible para todas estas universidades.

Identificación de entidades y recursos

Las entidades que compartirán información son:

Universidades

Los recursos a compartir son:

1) Contenido del curso
2) Material de apoyo del curso

4.2.3. Requerimientos

a) Tecnológicos

SOAP

La versión utilizada es SOAP 1.1.

Mediante este estándar se permitirá que las aplicaciones puedan transmitir sus datos por medio de sobres, que especificarán las condiciones de solicitud y respuesta a solicitudes de servicios.

UDDI

La versión utilizada es UDDI V3.0.

Para WSMB los servicios deberán ser publicados en forma privada, ya que solo las aplicaciones que estén registradas como miembros podrán tener acceso a los servicios ofrecidos.

WSDL

La versión utilizada es WSDL 1.1.

Por medio de WSDL se detallaran las descripciones de los servicios, que contendrán las especificaciones de la implementación y de las interfaces de los servicios.

b) De integración

Las universidades por tener las mismas características serán consideradas como un mismo componente. Se identificarán con el nombre de WSUniversidad y compartirán sus recursos. Es decir cualquier universidad comparte el contenido de los cursos y los materiales de apoyo sin restricciones y por otro lado podrán consultar cualquier información disponible en el WSU.

En la figura 4.13 se muestra el diagrama de integración, donde se muestran las dos interfaces donde se implementarán los servicios a compartir.

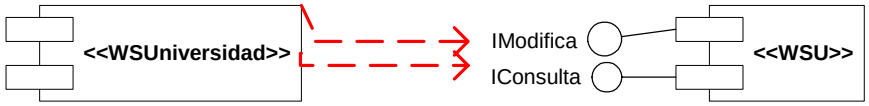


Figura 4.13. Diagrama de integración para la publicación de servicio

c) Orientados a servicios

Seguridad

Los profesores que deseen accesar a WSU deberán tener la certeza de que la información no podrá ser modificada por personas ajenas, también se deberá garantizar que la información disponible es originaria por cualquiera de las universidades involucradas.

Se implementará WS-S que es un estándar que proporciona especificaciones para garantizar la seguridad en Web Services.

Confiabilidad

Se esperará que cualquier información contenida realmente esté disponible, cuando algún profesor desee accesar a ella.

4.2.3.1. Casos de uso

Identificación de actores y casos de uso

A continuación se listan los actores y los casos de uso identificados.

Actor

AppUniversidad: representa a la aplicación correspondiente a cualquier universidad que desee integrar sus cursos.

Identificación de Casos de uso

Se clasifican en tres grupos:

a) Consulta cursos: mediante el cual un profesor podrá buscar los contenidos y los materiales de apoyo disponibles para formar un curso en base a sus preferencias (como se muestra en la figura 4.14).
   Selecciona contenidos
   Selecciona materia

b) Agrega servicios: ocurre cuando un profesor agregará un enunciado de contenido a una materia (ver figura 4.15).
   Agrega contenido
   Agrega materia

c) Creación de curso: es cuando un profesor forma su propio curso a partir de la información disponible (como se muestra en la figura 4.16).
   Integra curso
   Selecciona contenidos por materias
   Selecciona material de apoyo

4.2.3.2. Diagramas de casos de uso

Consulta cursos

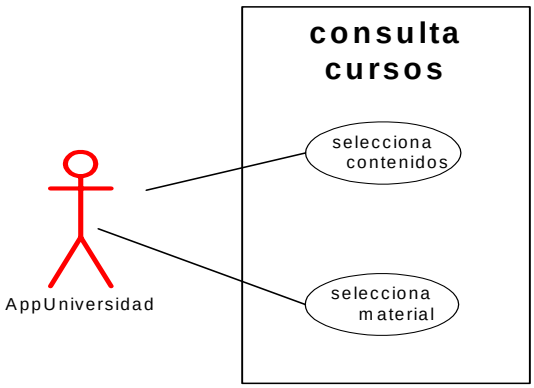


Figura 4.14. Diagrama del caso de uso “consulta cursos”

Agrega servicios

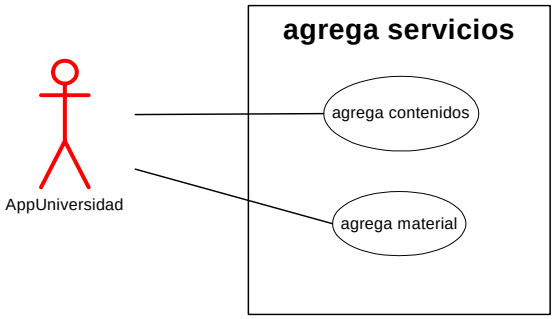


Figura. 4.15. Diagrama del caso de uso “agrega cursos”

Creación de curso

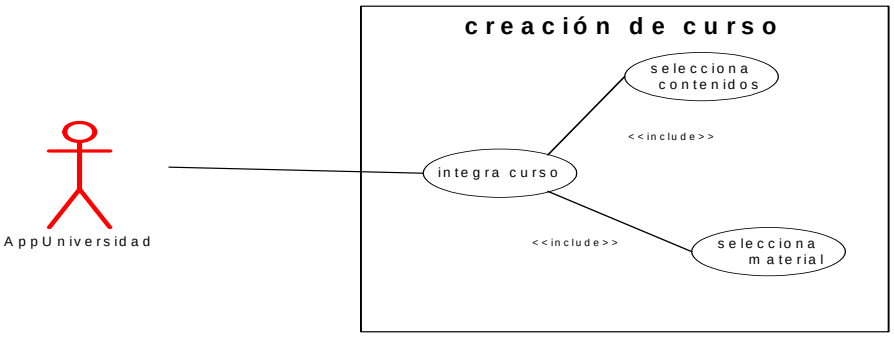


Figura 4.16. Diagrama del caso de uso “creación de curso”

4.2.3.3. Descripción de los casos de uso

A continuación se muestran las plantillas correspondientes a cada caso de uso definido.

Selecciona contenidos

ID: WSUcu1	VERSIÓN: 1.0may182003
NOMBRE:	Selecciona Contenidos
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppUniversidad	
DESCRIPCIÓN:	
Un AppUniversidad introducirá la clave de la materia para que el sistema le muestre todos los contenidos disponibles para esa materia. Después de que verifiqué que efectivamente existe la clave de la materia podrá seleccionar un contenido, seleccionando la clave del contenido.	
El sistema se encargará de verificar que el contenido exista.	

Figura. 4.17. Diagrama de la descripción “selecciona contenidos”

En la figura 4.17 se muestra la descripción textual del caso de uso “agrega cursos”, se identifica el actor involucrado y la narrativa del caso de uso.

La figura 4.18 muestra la descripción textual del caso de uso “selección material”, se identifica el actor que lo realiza.

La figura 4.19 muestra la descripción textual del caso de uso “agrega contenidos”, se identifica el actor que lo realiza.

La figura 4.20 muestra la descripción textual del caso de uso “agrega cursos”, identificando al actor que lo realiza.

En la figura 4.21 se muestra la descripción textual del caso de uso “integra curso”, se identifica el actor que lo realiza.

Selecciona material

ID: WSUcu2	VERSIÓN: 1.0may182003
NOMBRE:	Selecciona material
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppUniversidad	
DESCRIPCIÓN:	
Un AppUniversidad introducirá la clave de la materia para que el sistema le muestre el material disponibles para esa materia.	
El sistema verificará que efectivamente existe la clave de la materia.	
Y AppUniversidad podrá consultar el material	

Figura. 4.18. Diagrama de la descripción “selecciona material”

Agrega contenidos

ID: WSUcu3	VERSIÓN: 1.0may182003
NOMBRE:	Agrega contenidos
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppUniversidad	
DESCRIPCIÓN:	
Un AppUniversidad introducirá la clave de la materia para que el sistema verifique que la materia existe. Entonces podrá agregar un nuevo contenido y dejarlo disponible para el resto de los usuarios (aplicaciones).	
Cada contenido tendrá su propia clave de identificación.	

Figura. 4.19. Diagrama de la descripción “agrega contenidos”

Agrega material

ID: WSUcu4	VERSIÓN: 1.0may182003
NOMBRE:	Agrega material
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppUniversidad	
DESCRIPCIÓN:	
Un AppUniversidad introducirá la clave de la materia para que el sistema verifique que la materia existe. Entonces podrá agregar nuevo material y dejarlo disponible para el resto de los usuarios (aplicaciones). El material será accesible por la clave de la materia y por la clave del contenido.	

Figura. 4.20. Diagrama de la descripción “agrega material”

Integra curso

ID: WSUcu5	VERSIÓN: 1.0may182003
NOMBRE:	Formar curso
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppUniversidad	
DESCRIPCIÓN:	
Un AppUniversidad introducirá la clave de la materia para que el sistema verifique que la materia existe. Una vez que el sistema verifique la materia existe. AppUniversidad podrá consultar y utilizar toda la información disponible para esa materia, y por medio de un proceso de "selecciona contenidos" y "selecciona material", podrá formar su propio curso. El curso podría ser almacenado en WSU o bien simplemente creado por AppUniversidad exclusivamente para su uso.	

Figura. 4.21. Diagrama de la descripción “formar cursos”

4.2.3.4. Análisis de robustez

Los casos de uso fueron clasificados en tres grupos: consulta cursos, agrega servicios y creación de cursos. Estos diagramas identifican los objetos que intervienen en los casos de uso definidos en la sección anterior.

Selecciona contenidos

La figura 4.22 muestra el curso de acción cuando el actor AppUniversidad selecciona un contenido para lo cual introduce la clave de la materia representado por el objeto interface, la clave es evaluada por un objeto controlador y si es válida se despliega una lista de contenidos representada por un objeto entidad, finalmente el actor seleccionará el grupo de contenido de su elección que en la figura se representa con un objeto controlador. Si la clave no es válida a través de un objeto interface se le envía al actor un mensaje de error.

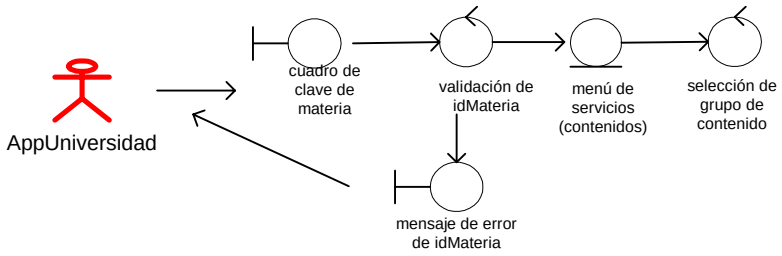


Figura. 4.22. Diagrama del análisis de robustez “selecciona contenidos”

Selecciona material

La figura 4.23 muestra el curso de acción cuando el actor AppUniversidad selecciona el material de apoyo para lo cual introduce la clave de la materia representado por el objeto interface, la clave es evaluada por un objeto controlador y si es válida se despliega una lista de los materiales disponibles representada por un objeto entidad, finalmente el actor seleccionará el material de su elección que en la figura se representa con un objeto controlador. Si la clave no es válida a través de un objeto interface se le envía al actor un mensaje de error.

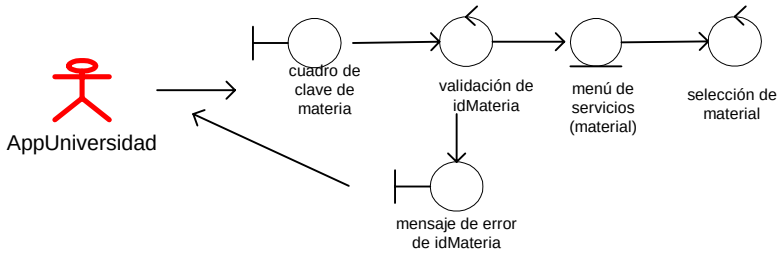


Figura. 4.23. Diagrama del análisis de robustez “selecciona material”

Agrega material

La figura 4.24 muestra el curso de acción cuando el actor AppUniversidad agrega material de apoyo para lo cual introduce la clave de la materia representado por el objeto interface, la clave es evaluada por un objeto controlador y si es válida se despliega el cuadro de agregar servicio representado por un objeto interface, en este caso se agrega el material si cumple con las especificaciones establecidas por las mismas universidades, si ha sido aceptado se agrega en el repositorio de cursos el cual es representado por un objeto entidad, si no es válido por medio de un objeto interface permite al actor corregir los datos y agregar el servicio otra vez. Si la clave no es válida a través de un objeto interface se le envía al actor un mensaje de error.

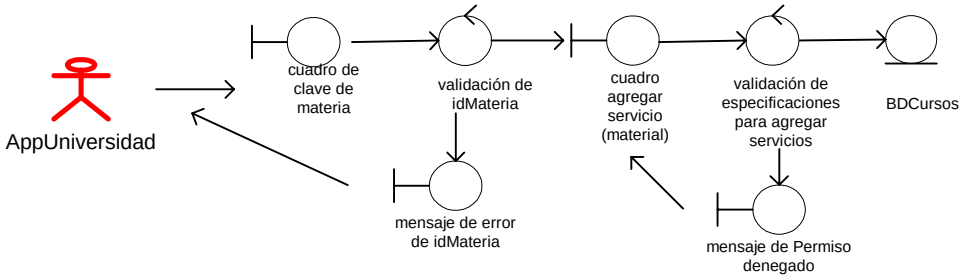


Figura. 4.24. Diagrama del análisis de robustez “agrega material”

Agrega contenidos

La figura 4.25 muestra el curso de acción cuando el actor AppUniversidad agrega contenidos para lo cual introduce la clave de la materia representado por un objeto interface, la clave es evaluada por un objeto controlador y si es válida se despliega el cuadro de agregar servicio representado por un objeto interface, en este caso se agrega el contenido si cumple con las especificaciones establecidas por las mismas universidades si ha sido aceptado se le asigna una clave, representado en la figura por medio de un objeto controlador y se agrega en el

repositorio de cursos el cual es representado por un objeto entidad, si no cumple con las especificaiciones por medio de un objeto interface se le permite al actor corregir los datos y agregar el servicio otra vez. Si la clave no es válida a través de un objeto interface se le envía al actor un mensaje de error.

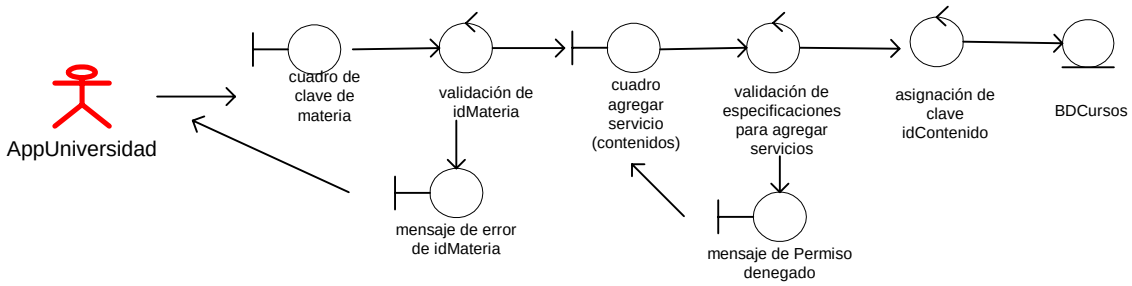


Figura. 4.25. Diagrama del análisis de robustez “agrega contenidos”

Integra cursos

La figura 4.26 se muestra el curso de acción cuando el actor AppUniversidad integra cursos para lo cual introduce la clave de la materia representado por el objeto interface, la clave es evaluada por un objeto controlador y si es válida se hace la invocación hacia el caso de uso “selecciona material” y “selecciona contenidos”, después que se han hecho estas selecciones, por medio del objeto interface “crear cursos” el repositorio del curso representado por una entidad recibe un mensaje de guardar curso, si el actor lo solicita. Si la clave no es válida a través de un objeto interface se le envía al actor un mensaje de error.

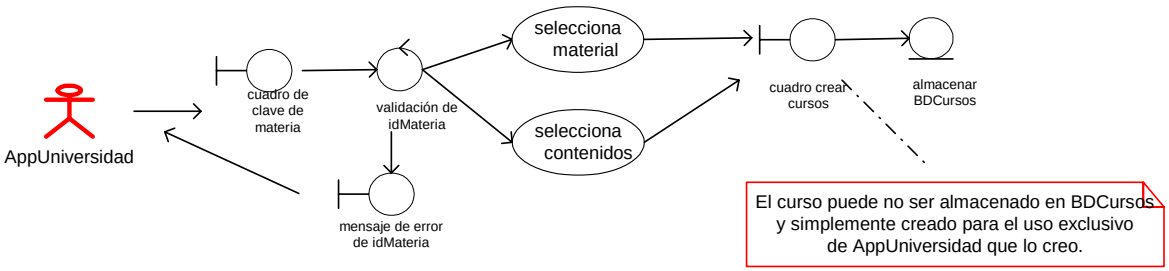


Figura. 4.26. Diagrama del caso de uso “integra cursos”

4.2.3.5. Diccionario del dominio

Se ha omitido la especificación del diccionario del domino porque el objetivo es validar la aplicabilidad de la modelación de los diagramas propuestos, pero esto

no significa que no sea importante realizarse. Para la creación de este diccionario se sugiere utilizar la plantilla propuesta en el capítulo 3.

4.2.4. Arquitectura

En las figuras 4.27 y 4.28 se muestran los diagramas para la vista de pila del proveedor y del solicitador o cliente, se observa que existe dependencia entre las capas de la pila de abajo hacia arriba, lo cual significa que la capa de abajo requiere de la capa de arriba para poder ser implementada.

Vista de pila

En la figura 4.27 se muestra el diagrama correspondiente al diagrama de la vista de pila para el proveedor del servicio, cada capa es representada por paquetes y cada paquete tiene comentado las consideraciones que deben ser tomadas en cuenta para la implementación de la arquitectura.

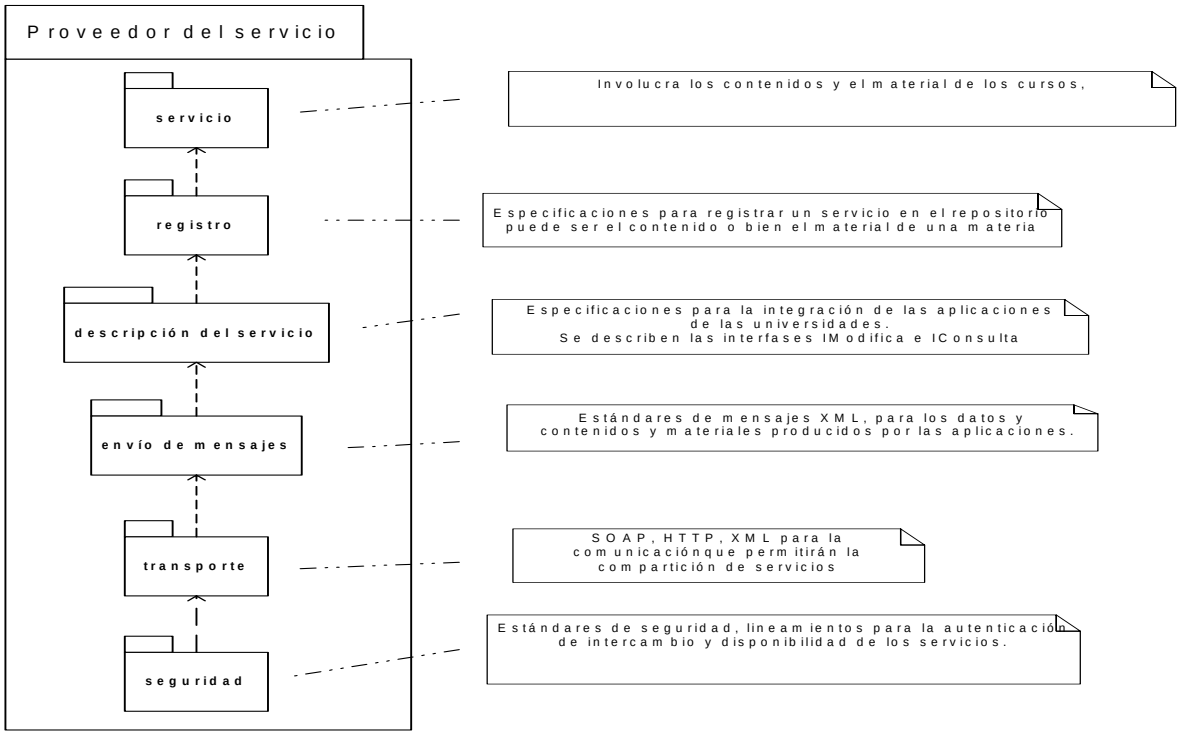


Figura 4.27. Diagrama de la vista de pila para el proveedor del servicio

Cliente

En la figura 4.28 se muestra el diagrama para la vista de pila del cliente, cada capa es representada por paquetes y cada una de ellas tiene comentados las especificaciones necesarias para la implementación de la arquitectura.

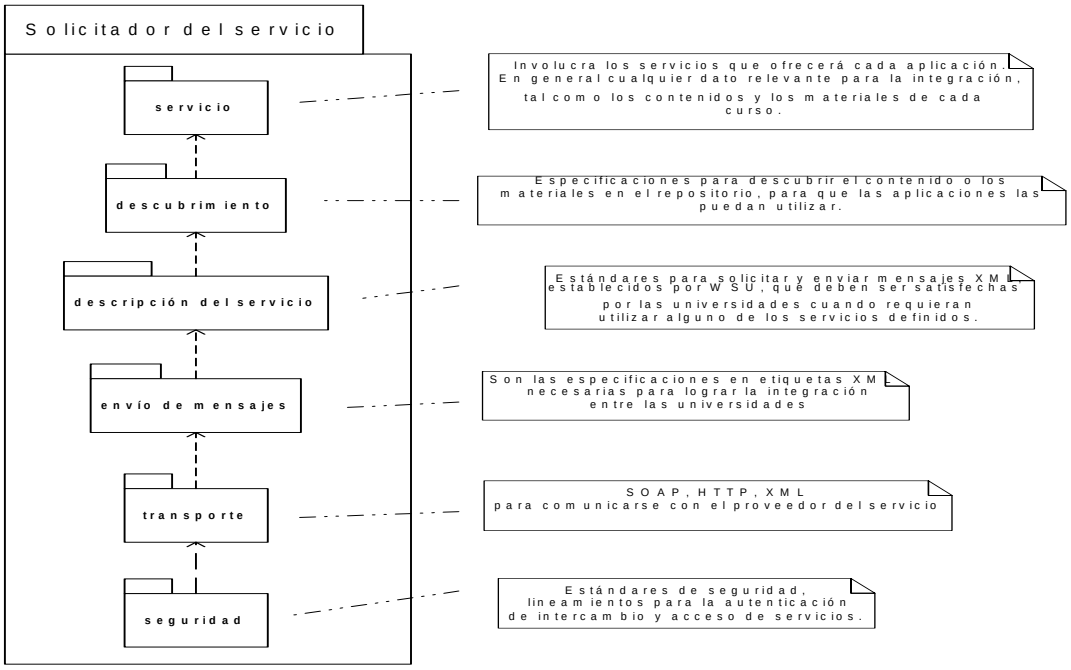


Figura 4.28. Diagrama de la vista de pila para el solicitador del servicio

Vista lógica

La figura 4.29 muestra la arquitectura en la vista de servicios (vista lógica) para la integración de servicios. En esta figura se representa la integración en forma lineal por el tipo de solicitud de servicio que se realiza, sin embargo es importante recordar que un WS Cliente puede solicitar servicios de igual forma que proveerlos. WSU se comunicará con el WS Cliente a través de dos interfaces Imodifica e Iconsulta, las cuales satisfacerán las solicitudes.

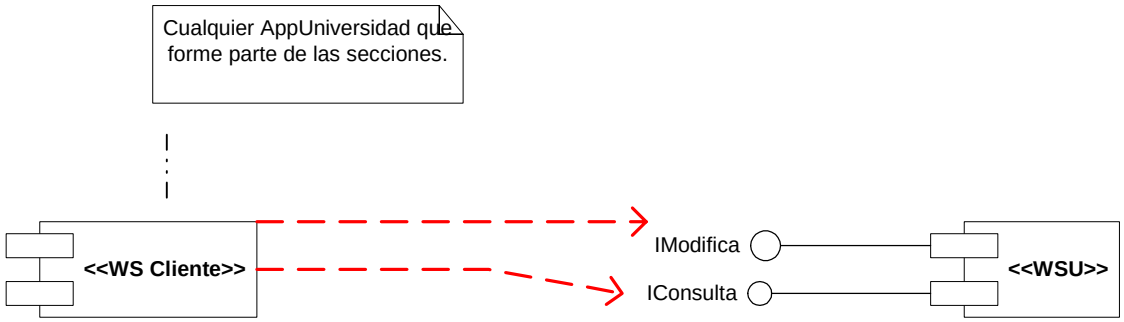


Figura 4.29. Diagrama de la vista lógica de la arquitectura

4.2.5. Análisis

4.2.5.1. Diagramas de clase

La figura 4.30 muestra el diagrama de clases para representación de WSU, se identifican las clases proveedor y solicitador de servicios y las interfaces que implementarán la compartición de procesos. Aunque para fines ilustrativos no se muestran atributos, solamente las asociaciones estereotipadas que representan los estándares en esta fase deben ser definidos y modelados.

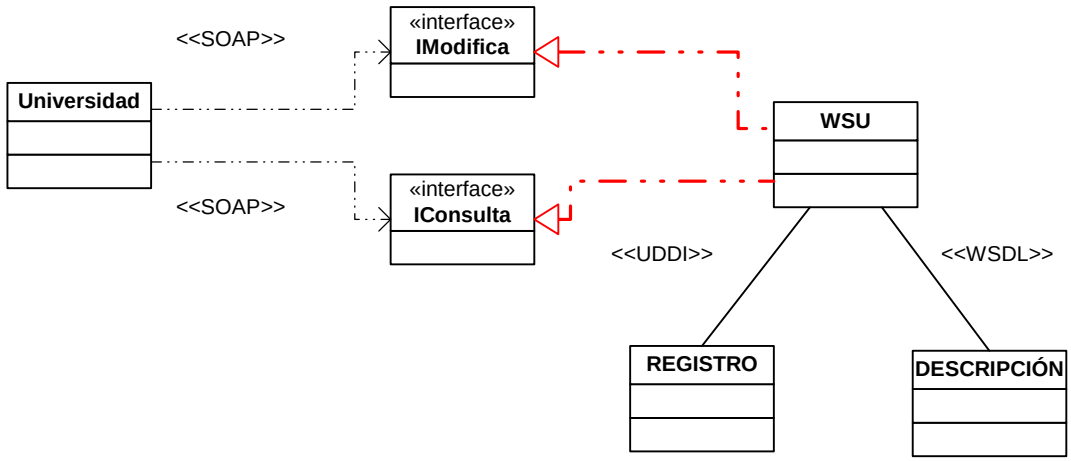


Figura 4.30. Diagrama de clases elaborado en la fase del análisis

4.2.5.2. Diagramas de secuencia

En las figuras siguientes figuras se exponen los diagramas de secuencia para cada diagrama del análisis de robustez elaborado en la fase de requerimientos.

Selecciona contenidos

La figura 4.31 muestra la secuencia de mensajes cuando el actor selecciona contenidos, en este diagrama el actor le envía al objeto interface la clave de la materia, la cual es enviada al controlador de la validación de la clave, si es válida el objeto interface le envía un mensaje de desplegar menú al objeto menú de servicios el cual permitirá la selección del servicio enviándole la solicitud del contenido específico. El controlador de la selección del servicio le enviará un mensaje al repositorio del curso representado por una entidad, si éste último objeto encuentra el contenido enviará de regreso el contenido solicitado. Si la clave de la materia no es válida el controlador de la validación le envía un mensaje al actor a través del objeto interface que representa la captura de la clave para que pueda ser introducida la clave correcta.

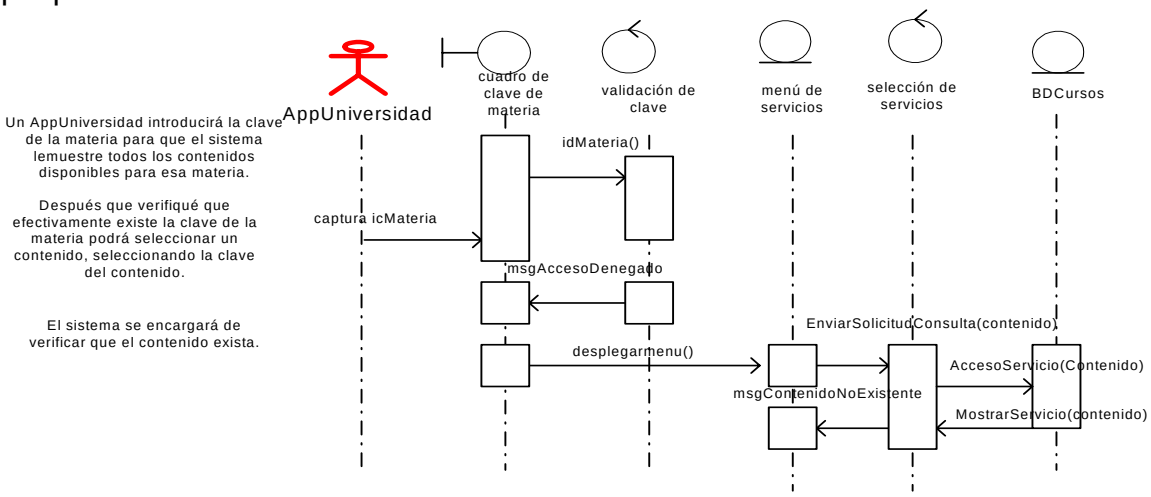


Figura 4.31. Diagrama de secuencia “selecciona contenidos”

Selecciona material

La figura 4.32 muestra la secuencia de mensajes cuando el actor selecciona el material de apoyo, en este diagrama el actor le envía al objeto interface la clave de la materia, la cual es enviada al controlador de la validación de la clave, si es válida el objeto interface le envía un mensaje de desplegar menú al objeto menú de servicios el cual permitirá la selección del servicio enviándole la solicitud del

material solicitado. El controlador de la selección del servicio le enviará un mensaje al repositorio del curso representado por una entidad, si éste último objeto encuentra el material le enviará de regreso el material solicitado. Si la clave de la materia no es válida el controlador de la validación le envía un mensaje al actor a través del objeto interface que representa la captura de la clave para que pueda ser introducida la clave correcta.

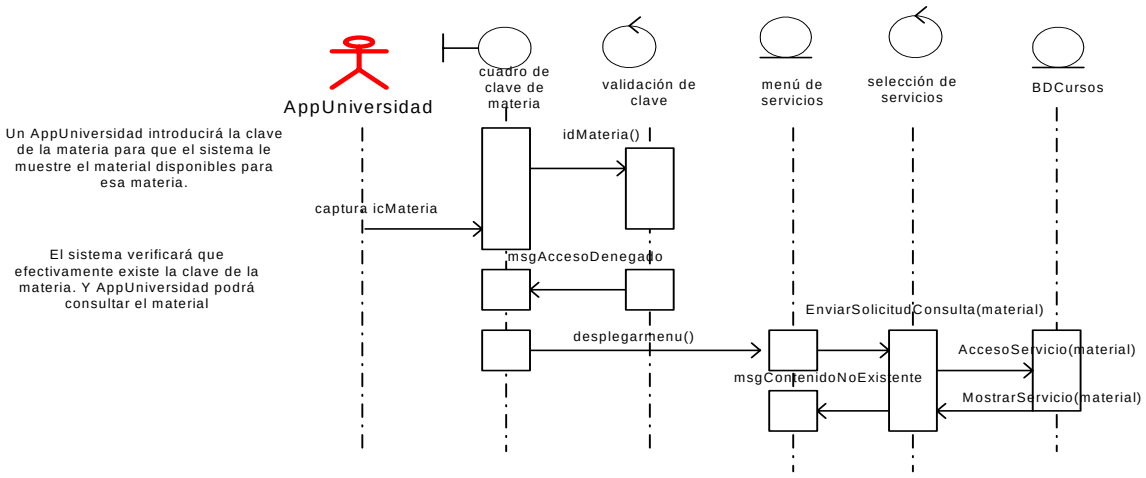


Figura 4.32. Diagrama de secuencia “selecciona material”

Agrega contenidos

La figura 4.33 muestra la secuencia de mensajes cuando el actor agrega contenidos, en este diagrama el actor le envía al objeto interface la clave de la materia, la cual es enviada al controlador de la validación de la clave, si es válida el objeto interface le envía un mensaje de agregar al objeto interface agregar servicio. El servicio que se desee agregar será evaluado para verificar que cumpla con las especificaciones establecidas por las universidades, este proceso es representado por el objeto validación que también tiene la tarea de asignar una clave al contenido, si cumple con las especificaciones y tiene asignada una clave por medio de un mensaje es agregado al repositorio de los cursos representado por un objeto entidad, éste último regresa al actor un mensaje de confirmación cuando ha sido agregado el servicio. Si la clave de la materia no es válida el controlador de la validación le envía un mensaje al actor a través del objeto interface que representa la captura de la clave para que pueda ser introducida la clave correcta.

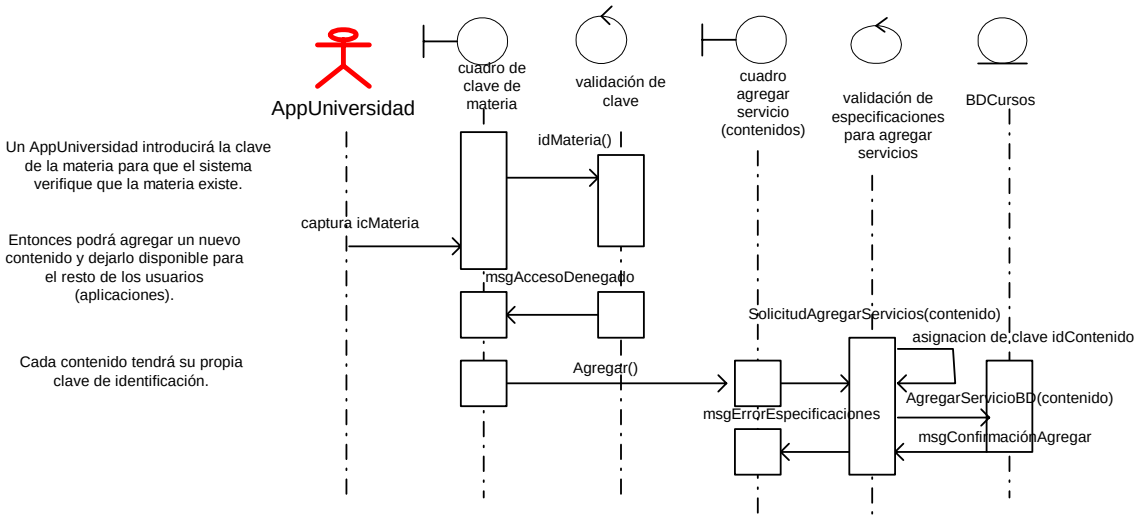


Figura 4.33. Diagrama de secuencia “agrega contenidos”

Agrega material

La figura 4.34 muestra la secuencia de mensajes cuando el actor agrega material de apoyo, en este diagrama el actor le envía al objeto interface la clave de la materia, la cual es enviada al controlador de la validación de la clave, si es válida el objeto interface le envía un mensaje de agregar al objeto interface agregar servicio. El servicio que se desee agregar será evaluado para verificar que cumpla con las especificaciones establecidas por las universidades, este proceso es representado por el objeto, si cumple con las especificaciones es agregado al repositorio de los cursos representado por un objeto entidad, éste último regresa al actor un mensaje de confirmación cuando ha sido agregado el servicio. Si la clave de la materia no es válida el controlador de la validación le envía un mensaje al actor a través del objeto interface que representa la captura de la clave para que pueda ser introducida la clave correcta.

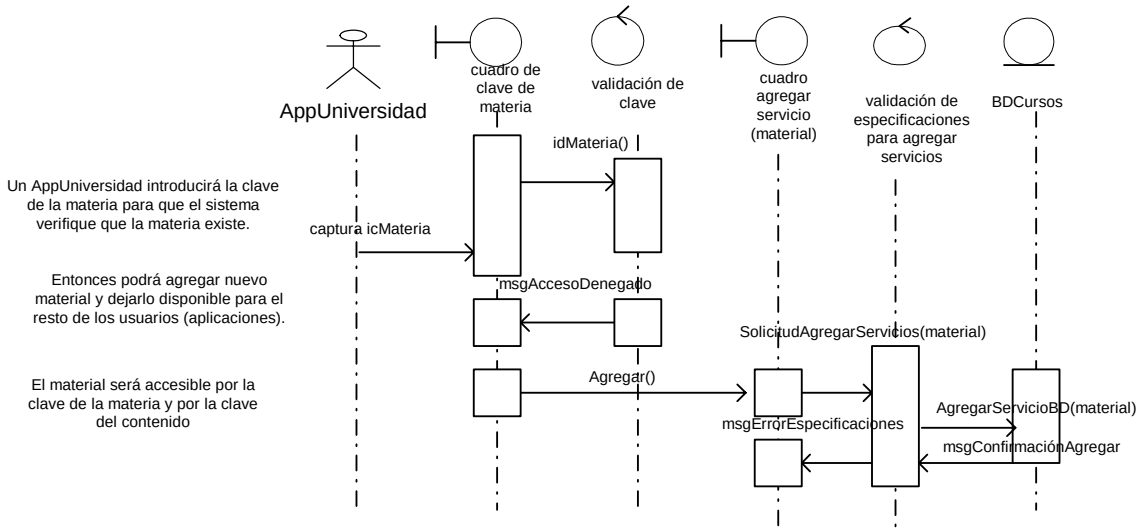


Figura 4.34. Diagrama de secuencia “agrega material”

Integra cursos

La figura 4.35 muestra la secuencia de mensajes cuando el actor integra cursos, en este diagrama el actor le envía al objeto interface la clave de la materia, la cual es enviada al controlador “validación de la clave”, si es válida se hace una invocación a seleccionar material y contenidos (figuras 4.32 y 4.33 respectivamente), después que se han seleccionado el objeto interface “crear cursos” el objeto interface le envía un mensaje de agregar al objeto interface “agregar servicio”. El servicio que se desee agregar será evaluado para verificar que cumpla con las especificaciones establecidas por las universidades, este proceso es representado por el objeto, si cumple con las especificaciones es agregado al repositorio de los cursos representado por un objeto entidad, éste último regresa al actor un mensaje de confirmación cuando ha sido agregado el servicio. Si la clave de la materia no es válida el controlador de la validación le envía un mensaje al actor a través del objeto interface que representa la captura de la clave para que pueda ser introducida la clave correcta.

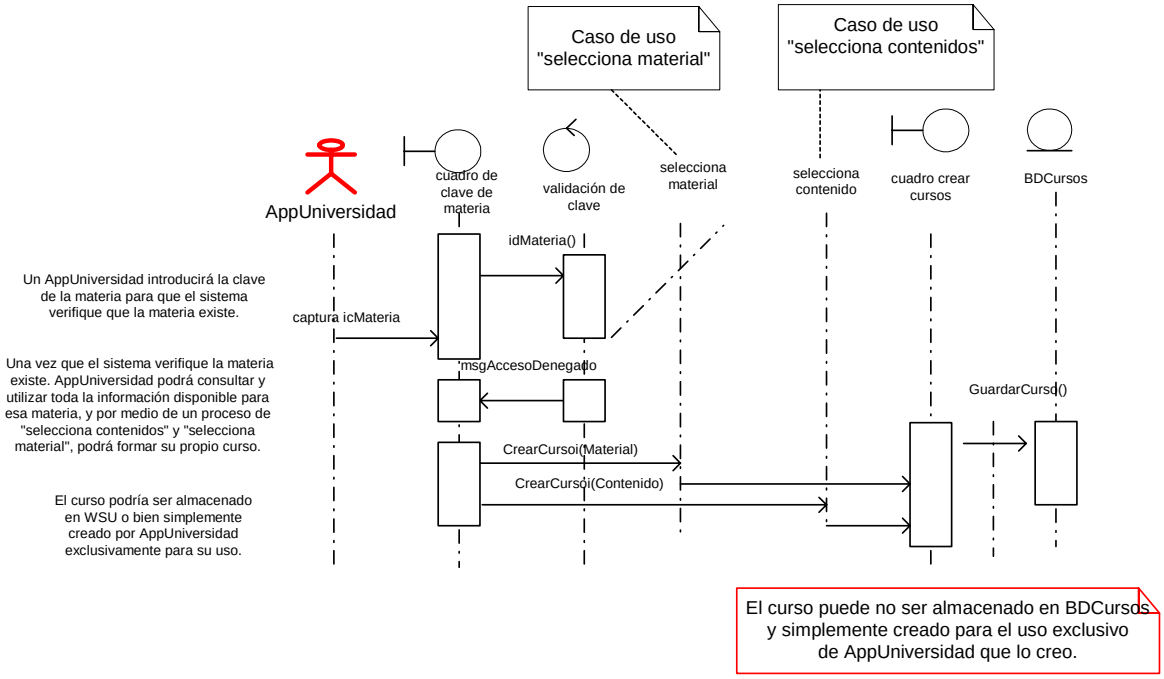


Figura 4.35. Diagrama de secuencia “integra cursos”

4.2.5.3. Diagramas de actividad

En la fase de identificación de casos de uso, éstos se agruparon en tres diferentes: consulta cursos, agrega servicios y creación de cursos. Estos diagramas permiten modelar la participación requerida por parte del proveedor y del solicitador de servicios.

Consulta cursos

La figura 4.36 muestra las actividades que debe realizar el cliente y el proveedor cuando se consultan cursos. El cliente (AppUniversidad) se comunica con el proveedor por medio de las interface “Iconsulta” para seleccionar un servicio al proveedor quien se encargará de obtener las descripciones de los servicios, los envía al cliente quien finalmente obtiene el servicio.

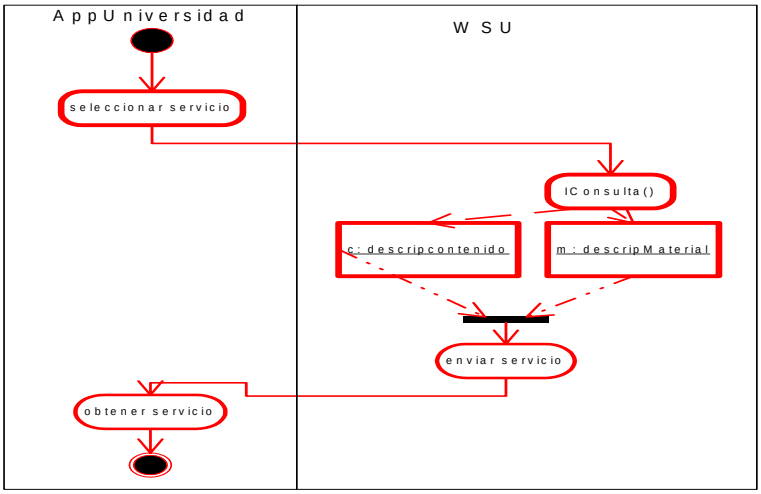


Figura 4.36. Diagrama de actividad “consulta cursos”

Agrega servicios

La figura 4.37 muestra las actividades que debe realizar el cliente y el proveedor cuando se agregan servicios. El cliente (AppUniversidad) genera una descripción dependiendo del servicio que desee agregar, esta descripción es enviada a WSU (proveedor) quien se encargará de validar y realizar el registro.

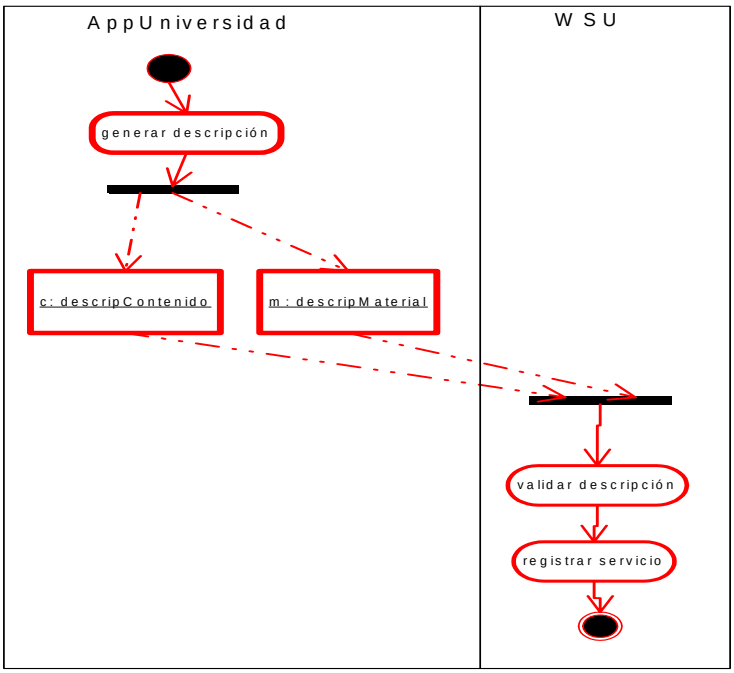


Figura 4.37. Diagrama de actividad “agrega servicios”

Creación de curso

La figura 4.38 muestra las actividades que debe realizar el cliente y el proveedor cuando se crea un curso. El cliente (AppUniversidad) realizar una selección del servicio obtiene las descripciones del proveedor las cuales son proporcionadas por medio de interfaces Iconsulta e Imodifica el cliente obtiene los servicios e integra el curso.

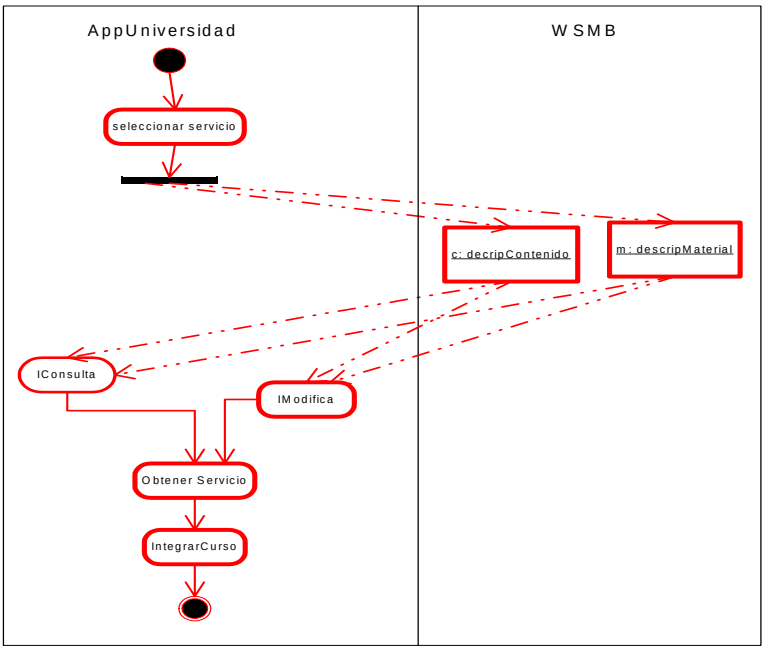


Figura 4.38. Diagrama de actividad “creación de curso”

4.2.6. Diseño

4.2.6.1. Diagramas de clases

La figura 4.39 muestra el diagrama de clases, el cual corresponde con el elaborado en la fase del análisis, aquí se identifican las operaciones que cada interface debe implementar y los métodos de las clases registro y descripción. Se observa como mediante relaciones estereotipadas se pueden identificar los estándares de Web Services.

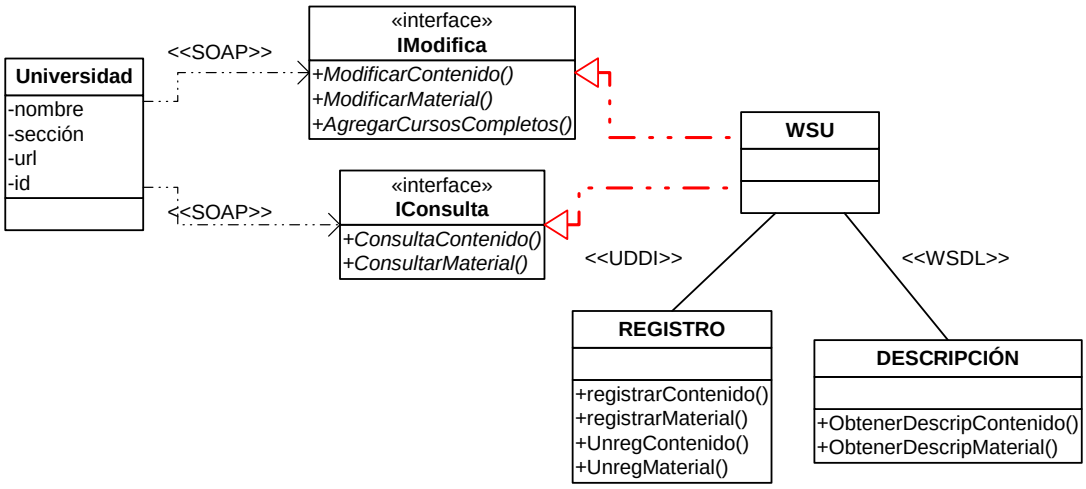


Figura 4.39. Diagrama de clases elaborado durante la fase de diseño

4.2.6.2. Diagramas de colaboración

En esta sección se muestran los diagramas de colaboración correspondientes a: consulta cursos, agrega servicios y creación de cursos.

Consulta cursos

Como se puede ver en la figura 4.40 la consulta de un curso inicia con la solicitud del servicio, esta solicitud es enviada a la interface “IConsulta” para poder obtener la descripción del servicio que se solicita al repositorio de cursos, el cual enviará un mensaje con el servicio solicitado.

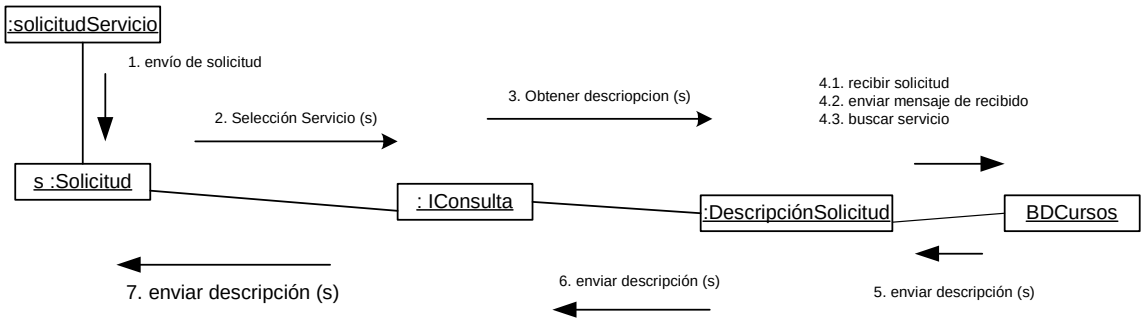


Figura 4.40. Diagrama de colaboración “consulta cursos”

Agrega servicios

Para agregar servicios primero se debe genera la descripción tal como se muestra en la figura 4.41, esta descripción será diferente si el servicio es un contenido o material de apoyo, la descripción es validad, si cumple con las especificaciones establecidas por las universidades, el servicio es registrado y se envía un mensaje de servicio registrado.

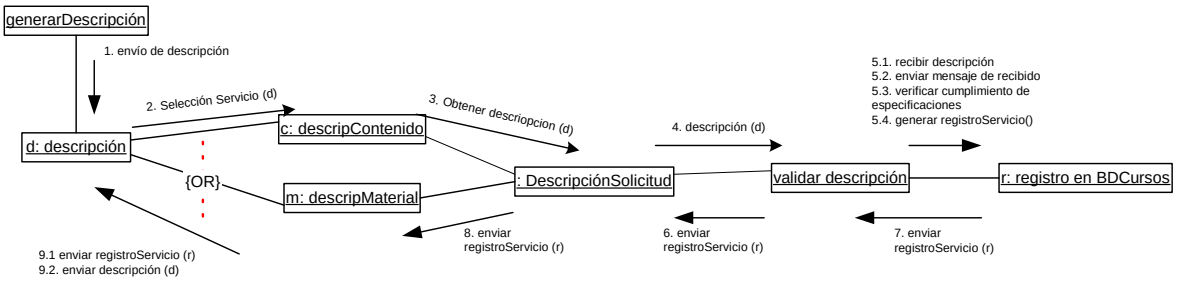


Figura 4.41. Diagrama de colaboración “agrega servicios”

Creación de curso

En la creación de un curso primero se selecciona el servicio para obtener el contenido o el material de apoyo, se obtienen sus respectivas descripciones y se genera la descripción tal como se muestra en la figura 4.42. Esta descripción será diferente si el servicio es un contenido o material de apoyo, la descripción es validada si cumple con las especificaciones establecidas por las universidades, el servicio es registrado y se envía un mensaje de “servicio registrado”.

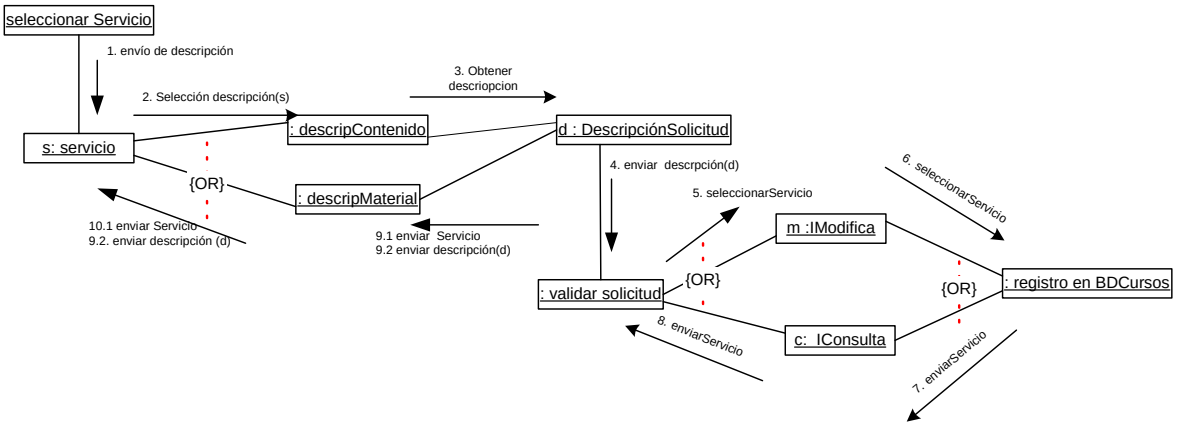


Figura 4.42. Diagrama de colaboración “creación de curso”

4.2.7. Codificación e Implementación y Pruebas

Estas fases han sido omitidas debido a que no se ha implementado todavía dicho sistema.

4.3 Resumen

En este capítulo se utilizó el proceso presentado en el Capítulo 3, aplicándolo a dos casos de estudio:

- Web Services MedBiquitous
- Web Services Universidades

Se omitieron algunos aspectos como el diccionario de datos porque se considera un documento técnico y el objetivo fue validar si los diagramas propuestos eran suficientes para la modelación de los casos de estudio, también se omitieron las fases de codificación e implementación y pruebas, debido a que en este trabajo no se llegó a la implementación de ninguno de ambos casos de estudio.

De la modelación de los casos de estudio, resultó que los diagramas resultaron convenientes para representar cada fase del ciclo de vida. A excepción del diagrama de clases para representar la arquitectura el cual no pudo ser aplicado ya que se carecían de especificaciones suficientes para poder ser aplicado en ambos casos de estudio.

En los dos siguientes capítulos se mostrarán los resultados y las conclusiones a las que se llegó, así como también los trabajos relacionados a este trabajo de tesis.

Capítulo 5 Conclusiones

En este capítulo se resumen los resultados obtenidos de la propuesta presentada en el capítulo 3, y validada en el capítulo 4. Se presenta una síntesis del trabajo realizado, los resultados obtenidos y las aportaciones.

Resumen del trabajo desarrollado

El proceso propuesto surge como un interés en proveer al desarrollador de sistemas de una guía para desarrollar Web Services, para lo cual fue necesario realizar un análisis del estado del arte en cuanto a metodologías de desarrollo de software se refiere; fueron analizadas cuatro de ellas:

- Larman [Lar1999]: “Análisis y diseño orientado a objetos”.
- Oestereich [Oes1999]: “Análisis y diseño orientado a objetos”.
- Rosenberg/Scott [RS1999]: “Análisis y diseño orientado a objetos”.
- Conallen [Con2000]: “Metodología de desarrollo de aplicaciones Web”.

Inicialmente fueron analizadas las metodologías mencionadas para conocer qué es lo que proponen en cada fase de desarrollo; conjuntamente se realizó un estudio acerca de los requerimientos de Web Services. Esto dio como resultado una perspectiva para identificar si alguna de estas metodologías satisface los requerimientos de Web Services o bien qué aspectos son considerados en algunas de las metodologías que pudieran ser útiles para modelar Web Services.

Después de obtener este resultado preliminar se procedió a proponer un proceso que cumpliera con tales requerimientos y además se tomaron los aspectos de las metodologías que aplican al desarrollo de Web Services, resultando en una proceso orientado a Web Services, expuesto en el capítulo 3 y que define la realización de dos documentos iniciales, titulados introducción y panorama general. Por otro lado sigue un desarrollo iterativo divido en requerimientos, arquitectura, análisis, diseño, codificación e implementación y pruebas.

Finalmente se valido este proceso mediante dos casos de estudio los cuales fueron expuestos en el capítulo 4.

5.1. Resultados

Los resultados obtenidos pueden ser sintetizados de la siguiente manera:

1. La propuesta de la representación de la arquitectura que propongo está basada en Oellerman [Oell2001] y en las diferentes pilas de Web Services propuestas por compañías desarrolladoras [Mye2001]. Sin embargo la representación de la arquitectura que propongo es diferente a la expuesta en tales documentos. Proponiendo una representación genérica, es decir, no depende de la arquitectura de ninguna compañía. Además la vista de servicio que propongo representar es una abstracción diferente a la arquitectura presentada por Oellerman.

2. Sugiero la utilización de plantillas para la elaboración de la descripción de los casos de uso y para la especificación del diccionario técnico. Estas plantillas resultaron prácticas durante el proceso de validación, porque ubica de manera rápida y sencilla los elementos que deben considerarse para la elaboración de tales documentos.

3. Inicialmente se propusieron algunos diagramas que en la práctica resultaron poco prácticos o bien no aplicables, esto llevó a modificar la propuesta hasta obtener el proceso presentado. También sucedió lo contrario, es decir, algunos diagramas no fueron propuestos desde el principio y se agregaron finalmente como resultado de un replanteamiento del proceso.

4. Todos los diagramas resultaron convenientes en ambos casos de estudio y fueron aplicados a excepción del diagrama de clases para modelar la arquitectura, debido a que se carecía de suficiente información para poder modelar alguno de los casos de estudio; sin embargo para otros sistemas es totalmente aplicable y relevante para identificar las dependencias entre la arquitectura.

5. Debido a que no se llegó a la implementación, no se pudo validar que algunos de los diagramas realizados en las fases anteriores pudieran ser utilizados en la fase de codificación e implementación.

5.2. Aportaciones

Las contribuciones realizadas pueden ser puntualizados de la siguiente manera:

1. Proponer un proceso para desarrollar Web Services, el cual es la aportación principal de este trabajo. Aunque existen otras propuestas de metodologías (ver capítulo 6), no encontré ninguna aplicable por completo al desarrollo de Web Services. Con esto tampoco quiero decir que mi propuesta no pudiera ser mejorada, sin embargo en el proceso de validación resultó adecuada, sencilla y práctica de aplicar. Es decir se obtuvo un proceso práctico que satisface los requerimientos de Web Services.

2. Identifiqué durante la etapa de validación algunas recomendaciones para realizar la modelación, las cuales son puntualizados en el capítulo 3.

3. En el proceso propuesto se destaca el hecho de que es el resultado del análisis de cuatro diferentes propuestas, lo cual es relevante en el sentido de que se toma la experiencia de otras personas que han trabajado en metodologías para proponer un proceso orientado completamente a Web Services.

4. Sugiero la utilización de estereotipos para representar los servicios como componentes estereotipados, la utilización de relaciones estereotipadas para representar los estándares que se requieren. También propuse la utilización de clases estereotipadas para la representación de la arquitectura.

Capítulo 6 Trabajos Relacionados

En este capítulo se presentan los trabajos relacionados encontrados, así como también posibles opciones de extensión a este trabajo.

Principalmente se tienen como trabajos relacionados las propuestas presentadas por Oestereich [Oes1999], Conallen [Con2000], Larman, [Lar1999] y Rosengerg/Scott [RS1999]. Todos ellos han trabajado en desarrollar metodologías de desarrollo de proyectos de software, quizás el que más se apegue al desarrollo de Web Services sea Conallen, sin embargo su propuesta está orientada completamente a aplicaciones Web.

Una propuesta presentada en [Arm2002] modela Web Services utilizando UML. La diferencia principal es que tal propuesta modela los estándares de Web Services, pero no considera la integración de aplicaciones tal como se hace en esta tesis. Otro trabajo relacionados es el desarrollado en [Car2001] para la modelación de UDDI utilizando UML. Estas dos propuestas se ocupan de la modelación de las especificaciones de los estándares Web Services solamente.

6.1. Trabajos a futuro

Posibles extensiones al proceso propuesto podrían ser:

- La inclusión de modelos de seguridad dentro de este proceso resultaría interesante. En el proceso presentado se sugiere la utilización de estándares, sin embargo proveer al desarrollador de un modelo a aplicar podría ser más beneficioso para el modelado.

- Podrían identificarse otras vistas de la arquitectura, esto debido a la evolución constante de los Web Services, por lo que resultaría interesante actualizar la representación de la arquitectura presentada.

- Dado que el objetivo de la tesis no era la implementación del Web Services, esta propuesta podría ser validada mediante el modelado para una aplicación que si se vaya a implementar físicamente y con lo cual podrían complementarse las fases de codificación e implementación y pruebas.

En relación al último punto es relevante mencionar que este proceso será utilizado por dos personas en su trabajo de tesis:

Alejandro García en “Modelo cooperativo para la administración del contenido de cursos en línea. Un enfoque basado en Web Services” [Gar2003].
Luis Francisco Canché Jiménez en “Implementación de Web Services en un sistema MES” [Can2003].

Anexo A "Datos biográficos de los autores de las metodologías analizadas"

A continuación se presentan algunos datos biográficos de los autores de las propuestas analizadas.

1. Conallen, Jim

Le llaman el "evangelista del modelado Web" dentro del Corporativo Rational Software, donde el ha trabajado en el desarrollo de WAE (Web Application Extension) –extensiones para aplicaciones Web utilizando UML-. Ha sido consultor especializado en el ciclo de vida de desarrollo Orientado a Objetos. Ha trabajado en el desarrollo de sistemas Web y cliente/servidor para el área de transportación, telecomunicaciones y salud. Ha diseñado y desarrollado aplicaciones en C++, Smalltalk, Java, Delphi y Visual Basic. Ha trabajado como arquitecto en sistemas de procesamiento de transacciones. Ha participado en conferencias sobre tópicos relacionados con el modelado de aplicaciones Web y, ha sido autor de diferentes publicaciones realizadas *en ASPToday.com, Rose Architect* y, *Communications of the ACM*.

2. Larman, Craig

Instructor principal en ObjectSpace (compañía especializada en la tecnología de agentes y objetos). Es creador de los patrones GRASP (General Responsability Asignment Software Pattern) –que tiene el propósito de capturar los principios fundamentales del diseño orientado a objetos-. Ha impartido cursos y asesorado a más de 2000 estudiantes.

3. Oestereich, Bernd

Hasta 1996 fue consultor y socio de *Putz & Partners Consulting*, posteriormente fue consultor líder para *debis Systemhouse* hasta 1998 cuando fundó su propia compañía *oose.de*. Es el autor de libros, artículos y seminarios sobre tecnología Orientada a Objetos.

4. Rosenberg, Doug

Fundador y presidente de ICONIX. Ha creado tutoriales sobre CORBA, UML y Unified Object Modeling Approach, los cuales han sido solicitados en más de 40 países. Ha colaborado con el modelado Orientado a Objetos con UML en compañías como *Boeing, Duke Power, Hughes, JC Penney, Lockheed-Martin, Lucent, Motorola, NASA* y, *Philip Morris*. Ha sido autor de artículos publicados en *Object Magazine, Software Development* y, Rose Architect.

5. Scott, Kendall

Es el autor de soporte de *UML Distilled* (Addison Weley Longman, 2ª. Edición 2000). Ha sido consultor y mentor en el modelado y captura de requerimientos usando un modelado de casos de uso. Es el director de *Software Documentation Wizards.*

Anexo B "Patrones GRASP"

En la siguiente tabla se presentan los patrones de diseño GRASP propuestos por Craig Larman [Lar1999].

Patrón	Descripción
experto	Se identifica la clase que posee la información necesaria para asumir la responsabilidad en el caso general.
creador	Asignar responsabilidad a la clase B de crear una instancia de clase A, sí: 1. B contiene a A 2. B agrega a A 3. B tiene los datos de inicialización de A 4. B registras a A 5. B utiliza a A
controlador	Asignar la responsabilidad de administrar un mensaje de eventos del sistema a una clase que represente una de las siguientes opciones: 1. El negocio o la organización global 2. El "sistema" global 3. Un ser animado del dominio que realice el trabajo 4. Una clase artificial que represente el caso de uso
bajo acoplamiento	Asignar responsabilidad de manera que se mantenga bajo acoplamiento.
alta cohesión	Asignar las responsabilidades de manera que se mantenga una alta cohesión.
polimorfismo	Cuando varía el tipo de comportamientos, asignar la responsabilidad del comportamiento a los tipos en que varía el comportamiento.
fabricación pura	Asignar un conjunto muy alto de cohesión de responsabilidades a una clase artificial que no represente nada en el dominio del problema, a fin de brindar soporte a una alta cohesión, a bajo acoplamientos y a la reutilización.
indirección	Asignar la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, de modo que no se acoplen directamente.
No hables con extraños	Asignar la responsabilidad al objeto directo del cliente para colaborar con el objeto indirecto, de manera que el cliente no necesita conocer el objeto indirecto.

Tabla B. Patrones GRASP (Larman)

Anexo C "Web Services Consorcio Medbiquitous WSMB"

Este anexo contiene el modelado completo del primer caso de estudio "Consorcio MedBiquitous" presentado en el capítulo 4. Este sistema fue modelado utilizando el proceso expuesto en el capítulo 3. A continuación se detallan los diagramas correspondientes a cada fase del proceso.

Caso de estudio:

"Consorcio MebBiquitous"

1. Introducción

Este caso de estudio fue obtenido de una sistema ya implementado, sin embargo en este trabajo se valida mediante el proceso propuesto en el capítulo anterior.

Situación

El Consorcio MedBiquitous es una organización no lucrativa fundada en mayo del 2001, creado por la Escuela de Medicina Johns Hopkins en asociación con otras sociedades médicas. Su misión es desarrollar tecnología para las comunidades relacionadas con la medicina que aporten educación profesional, experiencias laborales y, que colaboren para mejorar el nivel de salud en los pacientes. Para cumplir esta misión MedBiquitous ofrece una comunidad de comunidades, es decir, un portal integrado por comunidades médicas, sociedades que pueden ser formadas por médicos y/o por universidades y, entidades gubernamentales e industriales.

Solución

El portal desarrollado por MedBiquitous permite a los miembros trabajar juntos e intercambiar información de forma fiable, segura y automática. Este es una plataforma consistente de varias tecnologías integradas: XML permite a las

organizaciones involucradas mejorar el intercambio de datos, un framework en Java que permite incrementar la flexibilidad del software y mejorar el desarrollo, y Web Services que establece un canal de intercambio de información y de servicios disponibles por diferentes miembros a través de sus portales.

Beneficios

- La comunidad de comunidades desarrollada permite compartir información relevante de las últimas investigaciones realizadas por diferentes doctores que no están físicamente en el mismo lugar.

- El portal implementado facilita la capacitación continua, mediante los cursos en líneas disponibles y las publicaciones de *journals*.

- Utilizando este medio no lucrativo, los miembros pueden incrementar sus conocimientos y compartir el resultado de sus avances.

- Este portal define grupos de discusión disponibles en línea para todos los miembros.

- Un miembro puede participar en votaciones sobre políticas y estándares técnicos de Medicina.

2. Panorama general

Objetivo

Compartir resultados de investigaciones, *journals* y cursos en línea y participar en grupos de discusión mediante un portal confiable e independiente de plataformas de desarrollo.

Nombre

WSMB: Web Services MedBiquitous

Descripción de la aplicación

Mediante el desarrollo de Web Services se integrarán las aplicaciones de diferentes socios (comunidades, sociedades y entidades) de MedBiquitous. A través del Web Service (que para futuras referencias indistintamente será llamado portal), los miembros registrados podrán obtener los beneficios especificados en la sección anterior.

En cuanto a los estándares necesarios se han identificado la utilización de XML, Framework en Java y los estándares tecnológicos propios de Web Services.

Identificación de entidades y recursos

Se identifican tres entidades principales: WSMB recibirá información desde

4) sociedades (de profesionistas médicos o de universidades médicas),
5) organizaciones gubernamentales y/o industriales (p.e. hospitales y laboratorios)
6) otras comunidades médicas.

Los recursos que se desearán compartir son:

5) journals
6) cursos en línea
7) grupos de discusión
8) resultados de investigaciones

3. Requerimientos

3.1. Tecnológicos

SOAP
La versión utilizada es SOAP 1.1.

Mediante este estándar se permitirá que las aplicaciones puedan transmitir sus datos por medio de sobres, que especificarán las condiciones de solicitud y respuesta a solicitudes de servicios.

UDDI
La versión utilizada es UDDI V3.0.

Para WSMB los servicios deberán ser publicados en forma privada, ya que solo las aplicaciones que estén registradas como miembros podrán tener acceso a los servicios ofrecidos.

WSDL
La versión utilizada es WSDL 1.1.

Por medio de WSDL se detallaran las descripciones de los servicios, que contendrán las especificaciones de la implementación y de las interfaces de los servicios.

3.2. Integración

A continuación se especifican los servicios aportados por cada entidad:

- Una comunidad podrá publicar *journals* y resultados de investigaciones, ofrecer cursos en línea, formar y administrar grupos de discusión.

- Una sociedad podrá publicar *journals* y resultados de investigaciones y, ofrecer cursos en línea.

- Una organización gubernamental/industrial podrá publicar resultados de investigaciones y ofrecer cursos en línea.

WSMB recibirá solicitudes a servicios desde diferentes sociedades, comunidades u organizaciones industriales y gubernamentales, aún cuando algunas solicitudes pudieran coincidir estas serán específicas por lo que requerirán también de respuestas específicas. La figura C.1 muestra las interfaces que estarán implementadas en WSMB y la dependencia que existirá con respecto a cada entidad; en este diagrama se modelan los servicios que cada entidad aportará.

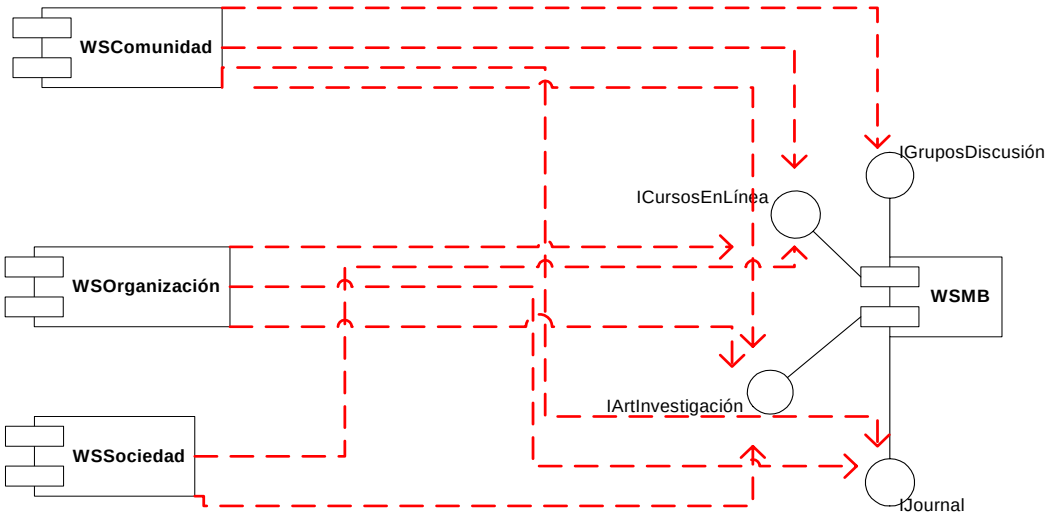


Figura C.1. Diagrama de integración para WSMB

3.3. Orientados al servicio

Seguridad

Los miembros que accedan a WSMB, podrán tener la certeza de que su información no será descubierta por otras personas que pudieran hacer mal uso de sus datos, por otro lado, tendrán acceso privado a los servicios, para lo cual será necesario proveer nombre de usuario y clave a cada miembro, para las entidades cliente.

Por otro lado WSMB autenticará el origen de cada servicio, además de garantizar que la información enviada no podrá ser interceptada por intrusos. También verificará que las entidades quienes estén ofreciendo un servicio cuenten con los requerimientos necesarios para hacerlo.

Una tecnología que podría ser implementada es HTTPS, que combina HTTP Basic Authentication (HTTP-AUTH) con Secure Socket Layer (SSL). HTTP-AUTH autoriza el acceso mediante una clave codificada con Base64; sin embargo por sí sólo no es un método seguro por lo que es necesario combinarlo con SSL para hacer un método más fuerte en cuanto a seguridad se refiere. SSL satisface los requerimientos de confidencialidad, autenticación e integridad.

Confiabilidad

Se esperará que cualquier miembro pueda realmente acceder a los servicios definidos en secciones anteriores. Es decir evitar que se tengan ligas a journals,

cursos en línea, grupos de discusión o publicaciones de resultados de investigaciones no disponibles.

3.4. Casos de uso

Identificación de los actores y casos de uso

En la definición de los casos de uso, la primera actividad es la identificación de los actores, que son todas aquellas entidades externas que interactuarán con el sistema.

Actores

- AppOrganización, representa a la aplicación correspondiente a las organizaciones.
- AppComunidad, representa a la aplicación correspondiente a las comunidades.
- AppSociedad, representa a la aplicación correspondiente a las sociedades.
- Administrador del portal, se encargará del registro de las aplicaciones.

Identificación de Casos de uso

Se agrupan los casos de uso de la manera siguiente:

e) Búsqueda de servicios. Cuando cualquier aplicación acceda al portal y requiera buscar uno de los recursos ofrecidos (figura C.2):
   Cursos en línea
   Consultar grupos de discusión
   Consultar *journals*
   Consultar artículos de investigación

f) Publicación de servicios. Cuando cualquier aplicación desee poner a disposición del resto de la comunidad de comunidades algún recurso (figura C.3):
   Ofrecer cursos en línea
   Crear grupos de discusión
   Agregar *journals*
   Agregar artículos de investigación

g) Uso de servicios. Cuando cualquier aplicación desee utilizar alguno de los recursos ofrecidos (figura C.4):
Participar en grupos de discusión
Obtener *journals* y artículos de investigación
Inscripción a cursos en línea

h) Registro de aplicaciones. Cuando una nueva aplicación quiera pertenecer a la comunidad de comunidades o bien, dejar de formar parte de ella (figura C.5):
Alta de aplicación
Baja de aplicación
Actualiza datos

3.5 Diagramas de casos de uso

La figura C.2 muestra el diagrama para los casos de uso en los que los actores AppComunidad, AppSociedad, AppOrganización desean invocar las consultas de los cursos en línea, grupos de discusión, journals o artículos de investigación, estos casos de uso fueron clasificados como búsqueda de servicios. En la figura se observa que cualquier de los actores podrían realizar las invocaciones hacia la consulta de los servicios.

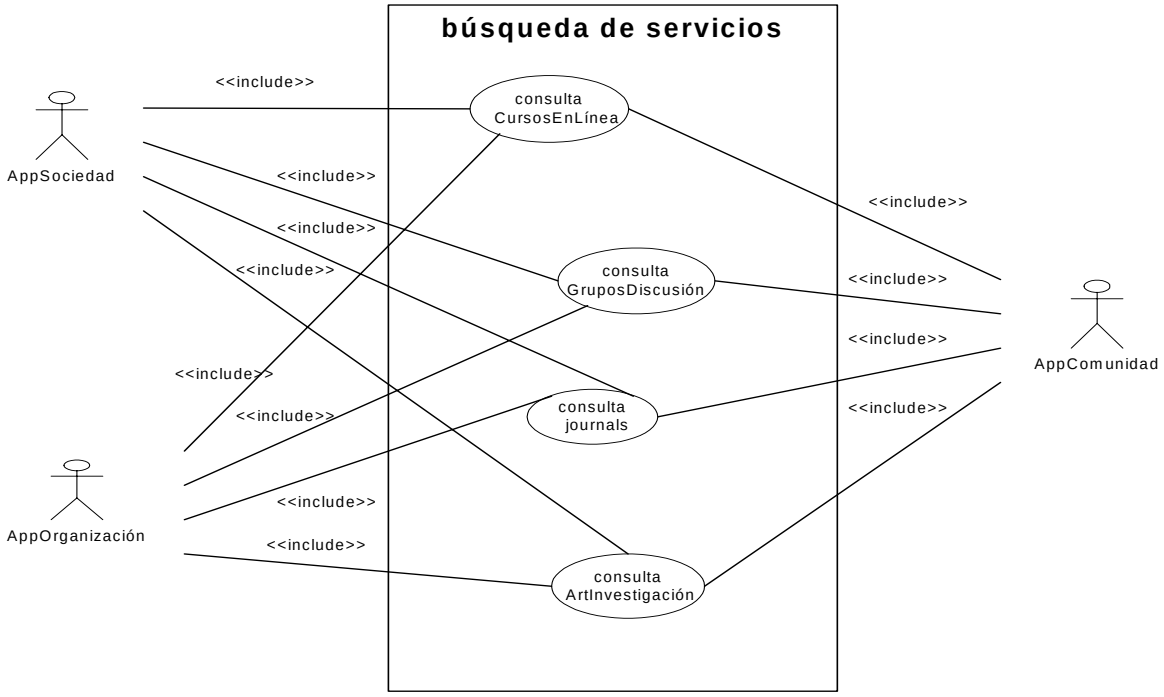


Figura C.2. Diagrama de caso de uso “Búsqueda de servicios”

La figura C.3 muestra el diagrama para los casos de uso en los que los actores AppComunidad, AppSociedad, AppOrganización desean ofrecer cursos en línea, crear grupos de discusión, agregar journals o artículos de investigación, estos casos de uso fueron clasificados como publicación de servicios. En la figura se observa que cualquier de los actores podrían realizar las invocaciones hacia la consulta de los servicios.

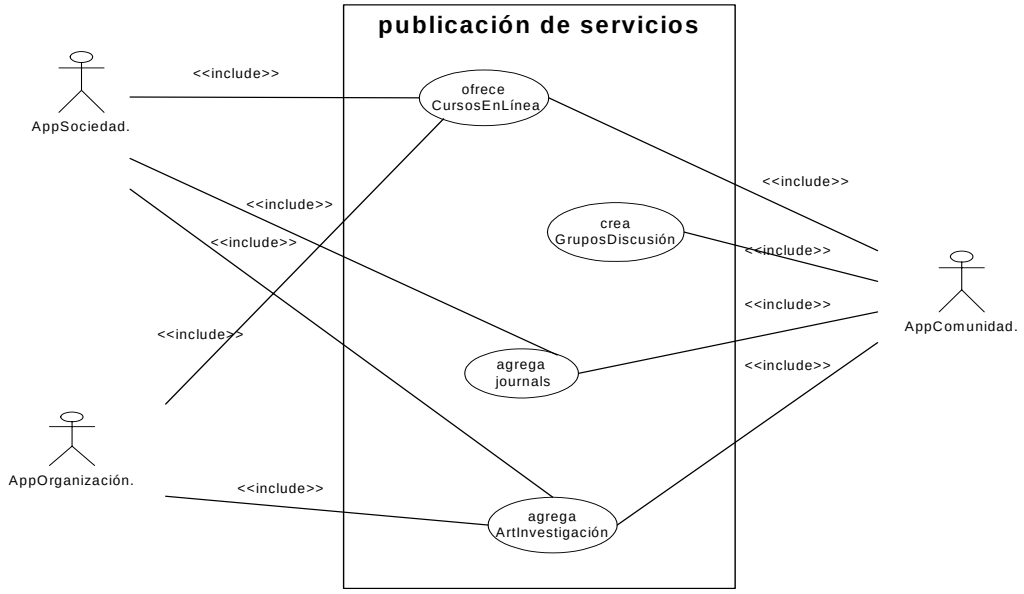


Figura C.3. Diagrama de caso de uso “Publicación de servicios”

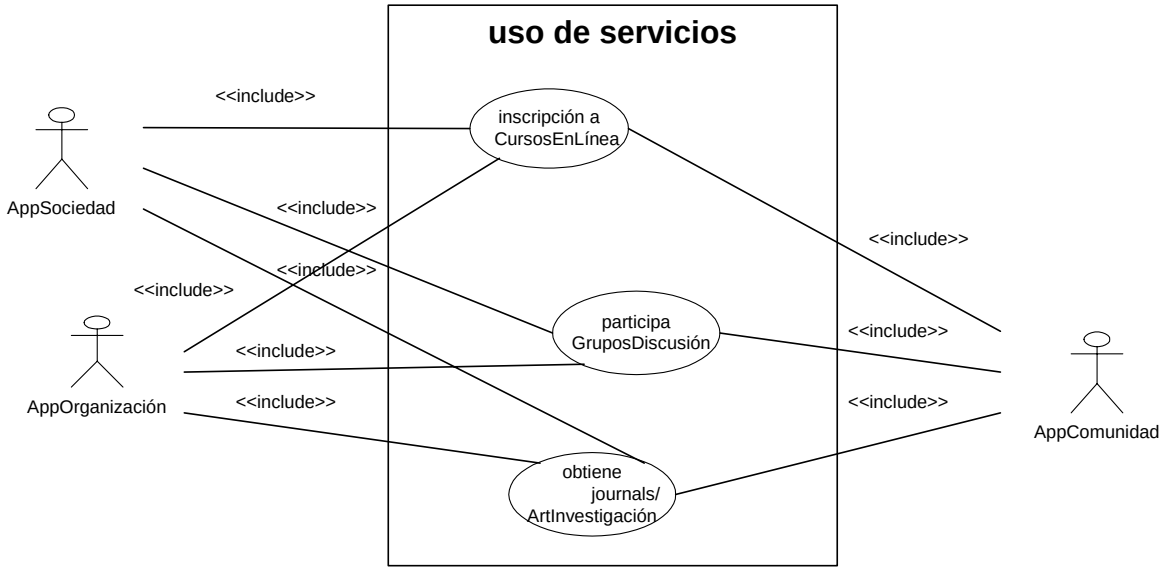


Figura C.4. Diagrama de caso de uso “Uso de servicios”

La figura C.4 muestra el diagrama para los casos de uso en los que los actores AppComunidad, AppSociedad, AppOrganización desean inscribirse a cursos en línea, participar en grupos de discusión u obtener journals o artículos de investigación, estos casos de uso fueron clasificados como uso de servicios. En la figura se observa que cualquier de los actores podrían realizar las invocaciones hacia la consulta de los servicios.

La figura C.5 muestra el diagrama para los casos de uso en los que los las aplicaciones deseen formar parte de la comunidad, será el administrador del portal quien las ingrese, las elimine de los registros o actualice los datos.

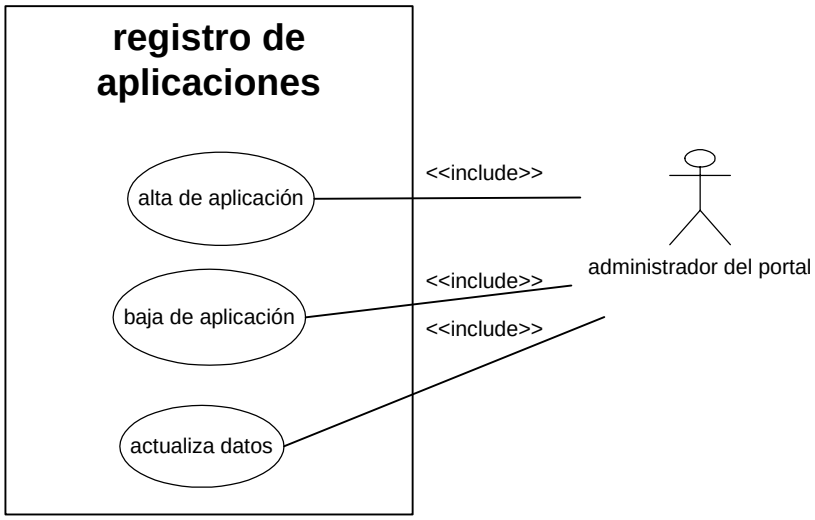


Figura C.5. Diagrama de caso de uso “Registro de aplicaciones”

3.6 Descripción de los casos de uso

En las siguientes plantillas se describe cada caso de uso definido.

Consulta cursos en línea

ID: WSMBcu1	VERSIÓN: 1.0may2003
NOMBRE:	Consulta CursosEnLínea
AUTORES:	MJLGL

PRECONDICIONES:
POSCONDICIONES:

ACTORES INVOLUCRADOS:
AppComunidad
AppSociedad
AppOrganización
DESCRIPCIÓN:
Un actor de cualquiera de los que están involucrados consultará un curso en línea, para lo cual accesará al portal por medio de una cuenta de usuario si es válido entonces podrá tener acceso a un menú de cursos en línea disponibles.

Consulta grupos de discusión

ID: WSMBcu2	VERSIÓN: 1.0may2003
NOMBRE:	Consulta GruposDiscusión
AUTORES:	MJLGL

PRECONDICIONES:
POSCONDICIONES:

ACTORES INVOLUCRADOS:
AppComunidad
AppSociedad
AppOrganización
DESCRIPCIÓN:
Un actor de cualquiera de los que están involucrados consultará la lista de los grupos de discusión, para lo cual proporcionará su cuenta de usuario si es válido entonces podrá tener acceso a un menú de grupos en línea y seleccionará el grupo que deseé consultar.
De ahí, podria hacer uso del grupo de discusión y participar.

Consulta *journals*

ID: WSMBcu3	VERSIÓN: 1.0may2003
NOMBRE:	Consulta Journals
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppComunidad,	
AppSociedad	
AppOrganización	
DESCRIPCIÓN:	
Un actor de cualquiera de los que están involucrados consultará la lista de journals publicados, para lo que debe registrarse por medio de una cuenta de ususario si es válido, seleccionará el journal de un menú de todos los journals disponibles.	

Consulta artículos de investigación

ID: WSMBcu4	VERSIÓN: 1.0may2003
NOMBRE:	Consulta ArtInvestigación
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppComunidad	
AppSociedad	
AppOrganización	
DESCRIPCIÓN:	
Un actor de cualquiera de los que están involucrados consultará la lista de artículos de investigación disponibles, para lo que deberá registrarse, si es válido, se le mostrará un menú de los artículos disponibles, de donde podrá seleccionar el que desee.	

Ofrece cursos en línea

ID: WSMBcu5	VERSIÓN: 1.0may2003
NOMBRE:	Ofrece CursosEnLínea
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	

ACTORES INVOLUCRADOS:
AppComunidad
AppSociedad
AppOrganización

DESCRIPCIÓN:
Un actor de cualquiera de los que están involucrados enviará una solicitud para ingresar un servicio (curso en línea), si cumple con los estándares establecidos por la comunidad de comunidades, entonces será agregado al repositorio, de lo contrario se le enviará mensaje de denegado el permiso para registrar servicios.

Crea grupos de discusión

ID: WSMBcu6	VERSIÓN: 1.0may2003
NOMBRE:	Crea GruposDiscusión
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	

ACTORES INVOLUCRADOS:
AppComunidad

DESCRIPCIÓN:
El actor AppComunidad enviará una solicitud para ingresar un servicio (grupo de discusión) , si cumple con los estándares establecidos por la comunidad de comunidades, entonces será agregado al foro de discusión, de lo contrario se le enviará mensaje de denegado el permiso para registrar servicios.
Si el permiso es denegado se le enviará un mensaje de error.

Agrega *journals*

ID: WSMBcu7	VERSIÓN: 1.0may2003
NOMBRE:	Agrega journals
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppComunidad	
AppSociedad	
DESCRIPCIÓN:	
El actor AppComunidad o AppSociedad enviarán una solicitud para ingresar un servicio (journals) , si cumple con los estándares establecidos por la comunidad de comunidades, entonces será agregado al repositorio de journals, de lo contrario se le enviará mensaje de denegado el permiso para registrar servicios.	

Agrega artículos de investigación

ID: WSMBcu8	VERSIÓN: 1.0may2003
NOMBRE:	Agrega ArtInvestigación
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	
AppComunidad	
AppSociedad	
AppOrganización	
DESCRIPCIÓN:	
Cualquier actor de los que están definidos como involucrados enviará una solicitud para agregar un servicio (ArtInvestigación) , si cumple con los estándares establecidos por la comunidad de comunidades, entonces será agregado al repositorio de artículos de investigación, de lo contrario se le enviará un mensaje de permiso denegado para registrar servicios.	

Participa en grupos de discusión

ID:	WSMBcu10 VERSIÓN: 1.0may2003
NOMBRE:	Participa GruposDiscusión
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	

ACTORES INVOLUCRADOS:

AppComunidad

AppSociedad

AppOrganización

DESCRIPCIÓN:

Cualquier actor de los que están definidos como involucrados accesará al portal, si su identificación es válida entonces se le mostrará un cuadro de aceptación y podrá accesar a un menú de servicios, donde seleccionarálos se el foro de discusión de su elección y podrá enviar o recibir corresos electrónicos.

Obtiene *journals* y artículos de investigación

ID:	WSMBcu11 VERSIÓN: 1.0may2003
NOMBRE:	Obtiene Journals/ArtInvestigación
AUTORES:	MJLGL

PRECONDICIONES:
POSCONDICIONES:

ACTORES INVOLUCRADOS:
AppComunidad
AppSociedad
AppOrganización
DESCRIPCIÓN:
Cualquier actor de los que están definidos como involucrados accesará
al portal, si su identificación es válida entonces se le mostrará un cuadro
de aceptación y podrá accesar a un menú de servicios, donde seleccionará
el journal o el artículo de investigación del respectivo repositorio.
Si el permiso es denegado se le enviar´aun mensaje de error de permiso.

Inscripción a cursos en línea

ID: WSMBcu9	VERSIÓN: 1.0may2003
NOMBRE:	Inscripción a CursosEnLínea
AUTORES:	MJLGL

PRECONDICIONES:
POSCONDICIONES:

ACTORES INVOLUCRADOS:
AppComunidad
AppSociedad
AppOrganización
DESCRIPCIÓN:
Cualquier actor de los que están definidos como involucrados accesará al portal, si su identificación es válida entonces se le mostrará un menú los servicios, de otra forma se le enviará un mensaje de acceso denegado. Una vez que ha accesado al menú, enviará una solicitud para inscribirse al curso en línea de su elección.

Alta de aplicación

ID: WSMBcu12	VERSIÓN: 1.0may2003
NOMBRE:	Alta de aplicación
AUTORES:	MJLGL

PRECONDICIONES:
POSCONDICIONES:

ACTORES INVOLUCRADOS:
Administrador del portal
DESCRIPCIÓN:
El administrador del portal dará acceso a toda aplicación que desee registrarse en el portal. El administrador del portal recibirá la solicitude por parte de la aplicación, verificará que el nombre de usuario y la clave no existan, si no existe se le enviará mensaje de bienvenida de lo contrario se le solicitará reingrese los datos.

Baja de aplicación

ID: WSMBcu13	VERSIÓN: 1.0may2003
NOMBRE:	Baja de aplicación
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	Administrador del portal
DESCRIPCIÓN:	El administrador del portal eliminará el registro de toda aplicación que desee dejar de participar en el portal. Para lo cual le solicitará un registro de acceso, es válido le mostrará una ventana para que solicité su baja del portal. Una vez eliminado el registro la aplicación recibirá un mensaje de confirmación de baja del portal.

Actualiza datos

ID: WSMBcu14	VERSIÓN: 1.0may2003
NOMBRE:	Actualiza datos
AUTORES:	MJLGL
PRECONDICIONES:	
POSCONDICIONES:	
ACTORES INVOLUCRADOS:	Administrador del portal
DESCRIPCIÓN:	El administrador del portal actualizará la información de las aplicaciones cuando estás lo soliciten. Para esto enviarán mediante un formulario los datos si cumplen con las especificacines establecidas por el portal se actualizarán los datos de otra forma se le volverán a solicitar sus datos.

3.7 Análisis de robustez

En la realización del análisis de robustez, se debe modelar un diagrama para cada caso de uso definido.

Consulta cursos en línea

La figura C.6 muestra el análisis de robustez para la consulta de cursos en línea, los actores (AppComunidad, AppOrganización, AppSociedad) inician la consulta por medio del objeto interface “cuadro de registro”, el cual después de que la clave de acceso al portal haya sido validada, muestra el menú de servicios representado por una entidad, de esta lista se selecciona un curso y se accede a ese curso a través de un registro. El curso se encuentra almacenado en el repositorio de “CursosEnLínea” representado por una entidad.

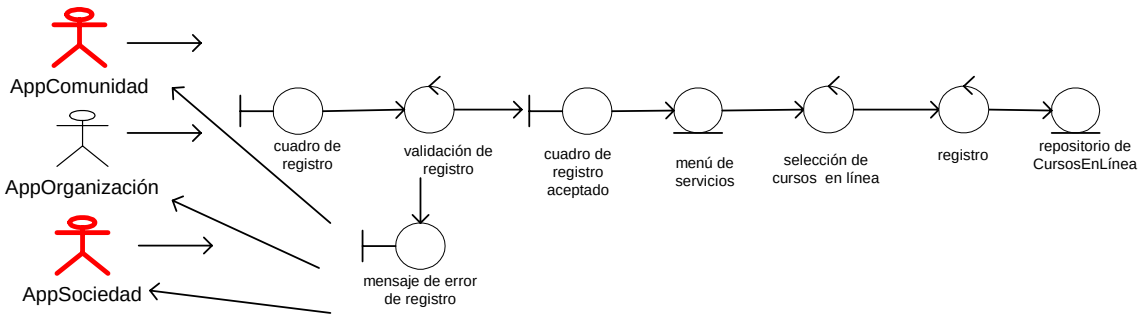


Figura C.6. Diagrama de análisis de robustez “Consulta cursos en línea”

Consulta grupos de discusión

La figura C.7 muestra el análisis de robustez para la búsqueda de servicios, en este diagrama se observa que un caso de uso puede estar relacionado con otro. El objeto interface “cuadro de registro” inicia el curso de la acción, después de que haya sido validada la clave de acceso al portal se desplegará el “menú de servicios” representado por una entidad, la validación se representa con el controlador “validación de registro”. El actor podrá seleccionar el grupo de discusión que desee consultar y eventualmente podría utilizar el caso de uso “participar grupo de discusión”.

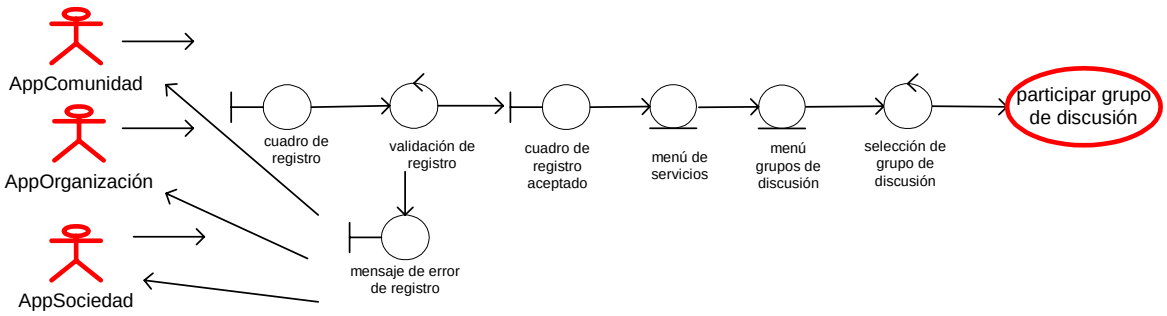


Figura C.7. Diagrama de análisis de robustez “Consulta grupos de discusión”

Consulta *journals*

La figura C.8 muestra el análisis de robustez para la consulta de *journals*, los actores (AppComunidad, AppOrganización, AppSociedad) inician la consulta por medio del objeto interface “cuadro de registro”, el cual después de que la clave de acceso al portal haya sido validada, muestra el menú de servicios representado por una entidad, de esta lista se selecciona un *journal*. El *journal* se encuentra almacenado en el repositorio de “*journals*” representado por una entidad.

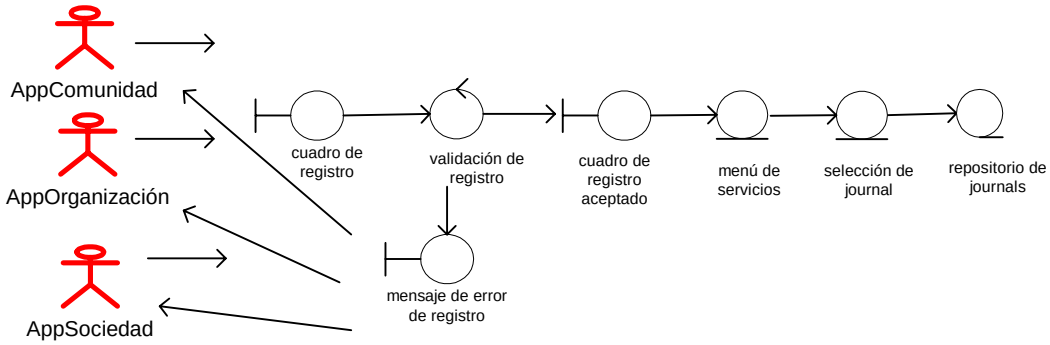


Figura C.8. Diagrama de análisis de robustez “Consulta *journals*”

Consulta artículos de investigación

La figura C.9 muestra el análisis de robustez para la consulta de artículos de investigación, los actores (AppComunidad, AppOrganización, AppSociedad) inician la consulta por medio del objeto interface “cuadro de registro”, después de que la clave de acceso al portal haya sido validada, muestra el menú de servicios representado por una entidad, de esta lista se selecciona un artículo de investigación. El artículo de investigación se encuentra almacenado en el repositorio de “artículos de investigación” representado por una entidad.

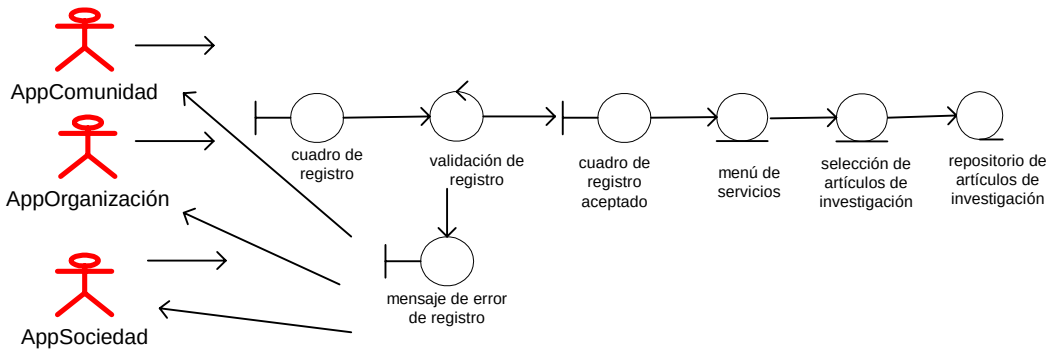


Figura C.9. Diagrama de análisis de robustez “Consulta artículos de investigación”

Ofrece cursos en línea

La figura C.10 muestra el análisis de robustez cuando se ofrecen cursos en línea, los actores (AppComunidad, AppOrganización, AppSociedad) inician la acción por medio del objeto interface “solicitud de ingreso de servicios”, después de que la solicitud haya sido validada se envía la solicitud al repositorio de “CusosEnLínea”. La validación como puede verse en el diagrama está representado por un controlador. Si la solicitud no cumple con las especificaciones se le envía al actor un mensaje de error por medio de un objeto interface.

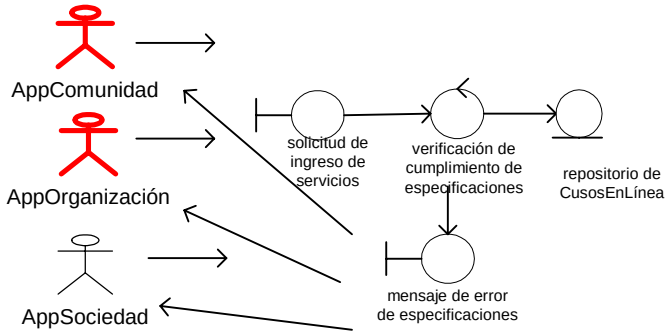


Figura C.10. Diagrama de análisis de robustez “Ofrece cursos en línea”

Crea grupos de discusión

La figura C.11 muestra el análisis de robustez cuando se crean grupos de discusión, los actores (AppComunidad, AppOrganización, AppSociedad) inician la acción por medio del objeto interface “solicitud de ingreso de servicios”, después de que la solicitud haya sido validada se envía la solicitud al repositorio de “Foro de discusión”. La validación como puede verse en el diagrama está representado por un controlador. Si la solicitud no cumple con las especificaciones se le envía al actor un mensaje de error por medio de un objeto interface.

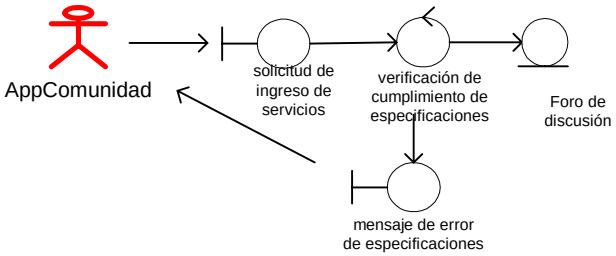


Figura C.11. Diagrama de análisis de robustez “Crea grupos de discusión”

Agrega *journals*

La figura C.12 muestra el análisis de robustez cuando se agregan *journals*, los actores (AppComunidad, AppOrganización, AppSociedad) inician la acción por medio del objeto interface “solicitud de ingreso de servicios”, después de que la solicitud haya sido validada se envía la solicitud al repositorio de “*journals*”. La validación como puede verse en el diagrama está representado por un controlador. Si la solicitud no cumple con las especificaciones se le envía al actor un mensaje de error por medio de un objeto interface.

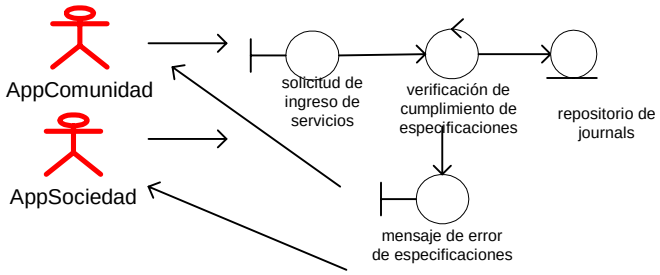


Figura C.12. Diagrama de análisis de robustez “agrega *journals*”

Agrega artículos de investigación

La figura C.12 muestra el análisis de robustez cuando se agregan artículos de investigación, los actores (AppComunidad, AppOrganización, AppSociedad) inician la acción por medio del objeto interface “solicitud de ingreso de servicios”, después de que la solicitud haya sido validada se envía la solicitud al repositorio “ArtInvestigación”. La validación como puede verse en el diagrama está representado por un controlador. Si la solicitud no cumple con las especificaciones se le envía al actor un mensaje de error por medio de un objeto interface.

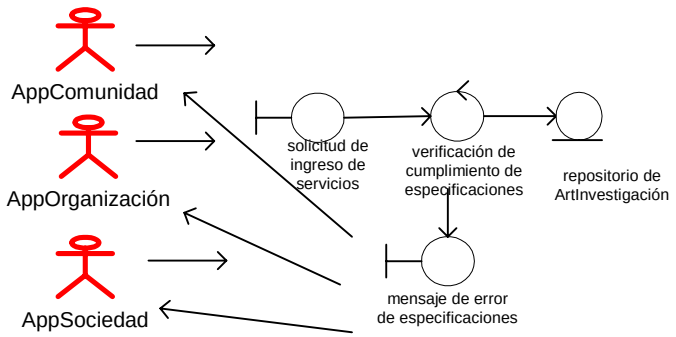


Figura C.12. Diagrama de análisis de robustez “agrega artículos de investigación”

Participa en grupos de discusión

La figura C.13 muestra el análisis de robustez cuando los actores (AppComunidad, AppOrganización, AppSociedad) participan en grupos de discusión, el inicio de la acción se representa con el objeto interface “cuadro de registro”, si es válido se muestra el menú de servicios que se representa por un objeto entidad, posteriormente se selecciona el foro de discusión representado por un objeto interface y se pueden enviar y recibir mail que llegan a la entidad final que es el “mailbox”. Si el registro no es válido se genera un mensaje de error y se permite que el actor intente nuevamente el acceso al portal.

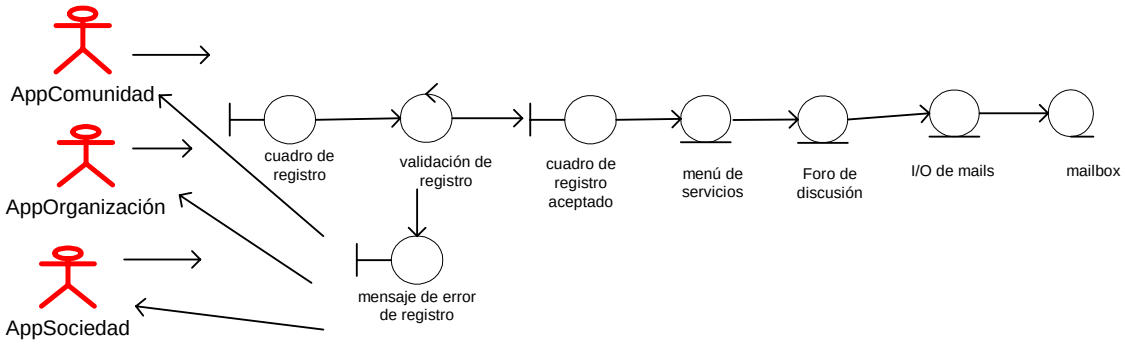


Figura C.13. Diagrama de análisis de robustez “Participa en grupos de discusión”

Obtiene *journals* y artículos de investigación

La figura C.14 muestra el análisis de robustez cuando los actores (AppComunidad, AppOrganización, AppSociedad) desean obtener *journals* y artículos de investigación, el inicio de la acción se representa con el objeto interface “cuadro de registro”, si es válido se muestra el menú de servicios que se representa por un objeto entidad, posteriormente se selecciona el servicio deseado si es un *journal* se obtendrá del “repositorio de *journals*”, si es un artículo de

investigación se obtendrá del repositorio “ArtInvestigación” los cuales son representado en el diagrama por entidades. Si el registro no es válido se genera un mensaje de error y se permite que el actor intente nuevamente el acceso al portal.

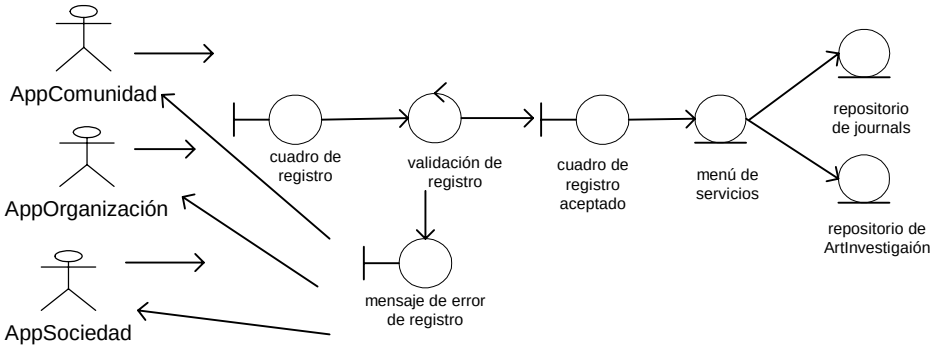


Figura C.14. Diagrama de análisis de robustez “Participa en grupos de discusión”

Inscripción a cursos en línea

La figura C.15 muestra el análisis de robustez cuando los actores (AppComunidad, AppOrganización, AppSociedad) se inscriben a cursos en línea, el inicio de la acción se representa con el objeto interface “cuadro de registro”, si es válido se muestra el menú de servicios que se representa por un objeto entidad, y se le solicita al actor llenar una solicitud de ingreso que se representa con una entidad, si es válida esta solicitud se envía al repositorio de “CursosEnLínea”. Si el registro no es válido se genera un mensaje de error y se permite que el actor intente nuevamente el acceso al portal.

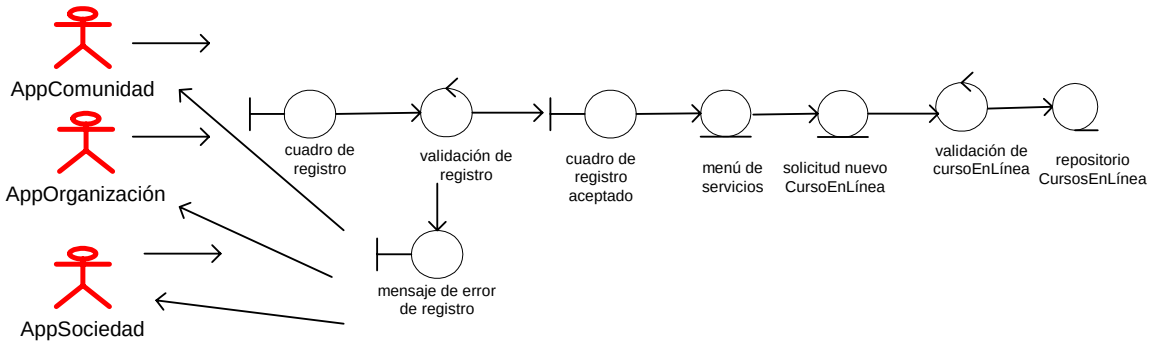


Figura C.15. Diagrama de análisis de robustez “Inscripción a cursos en línea”

Alta de aplicación

La figura C.16 muestra el análisis de robustez cuando el administrador del portal desea dar de alta una aplicación dentro de la comunidad, la acción inicia cuando el actor le envía al objeto interface “solicitud AltaAplicación”, si está correcta la solicitud verifica que la clave de acceso y el nombre de usuario proporcionados por el actor puedan ser válidos, ambas validaciones son representadas por controladores. Si resulta correcta esta última validación se ingresa a la base de datos representada por una entidad.

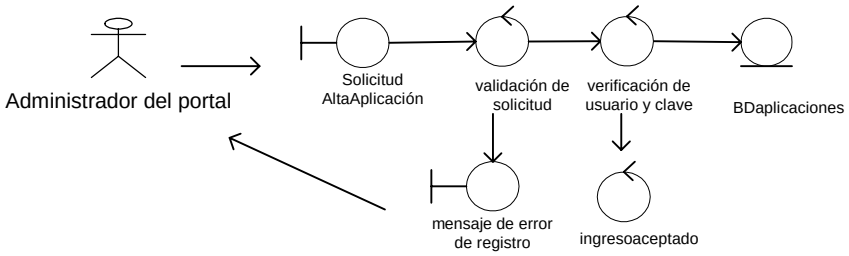


Figura C.16. Diagrama de análisis de robustez “Alta de aplicación”

Baja de aplicación

La figura C.17 muestra el análisis de robustez cuando el administrador del portal desea dar de baja una aplicación de la comunidad. La acción inicia cuando el actor valida la solicitud que recibe para dar de baja un registro, esta validación está representada por un controlador, si el registro es válido se muestra un objeto interfase “BajaAplicación” y envía un mensaje de confirmación, representado también por un objeto interface.

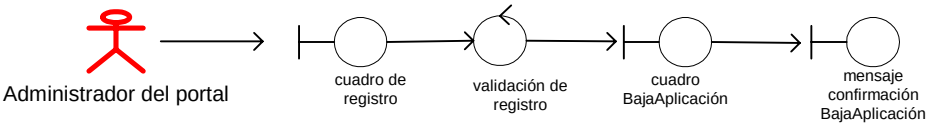


Figura C.17. Diagrama de análisis de robustez “Baja de aplicación”

Actualiza datos

La figura C.18 muestra el análisis de robustez cuando el administrador del portal desea dar de baja una aplicación de la comunidad. La acción inicia cuando el actor valida la solicitud que recibe para actualizar datos de un registro, esta validación está representada por un controlador, si el registro es válido se muestra un objeto interface “DatosAplicación”, se valida que los datos sean correctos y se ingresan los datos en la base de datos representada por una entidad.

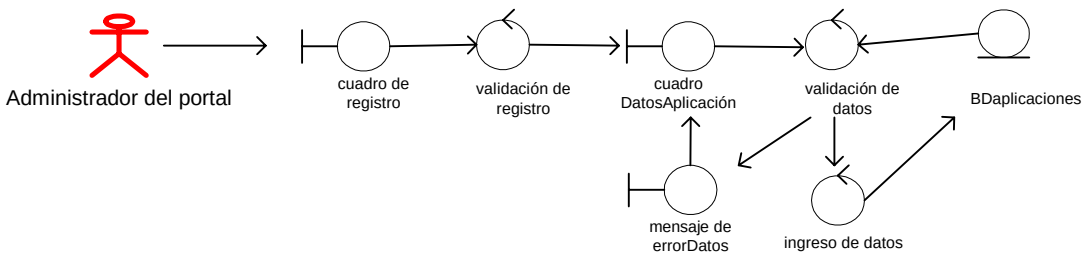


Figura C.18. Diagrama de análisis de robustez “Actualiza datos”

3.8 Diccionario del dominio

Se ha omitido en este documento sin embargo por ser un documento con fines técnicos y el propósito principal de este trabajo es demostrar la validez del proceso en el sentido de la aplicabilidad de los diagramas; sin embargo esto no significa que no sea importante o pueda no realizarse.

4. Arquitectura

4.1. Vista de pila

Servidor

La figura c.19 muestra la vista de pila para el proveedor del servicio, en la cual se identifican las seis capas de la arquitectura, cada capa tiene comentado los aspectos que deben ser considerados para su implementación.

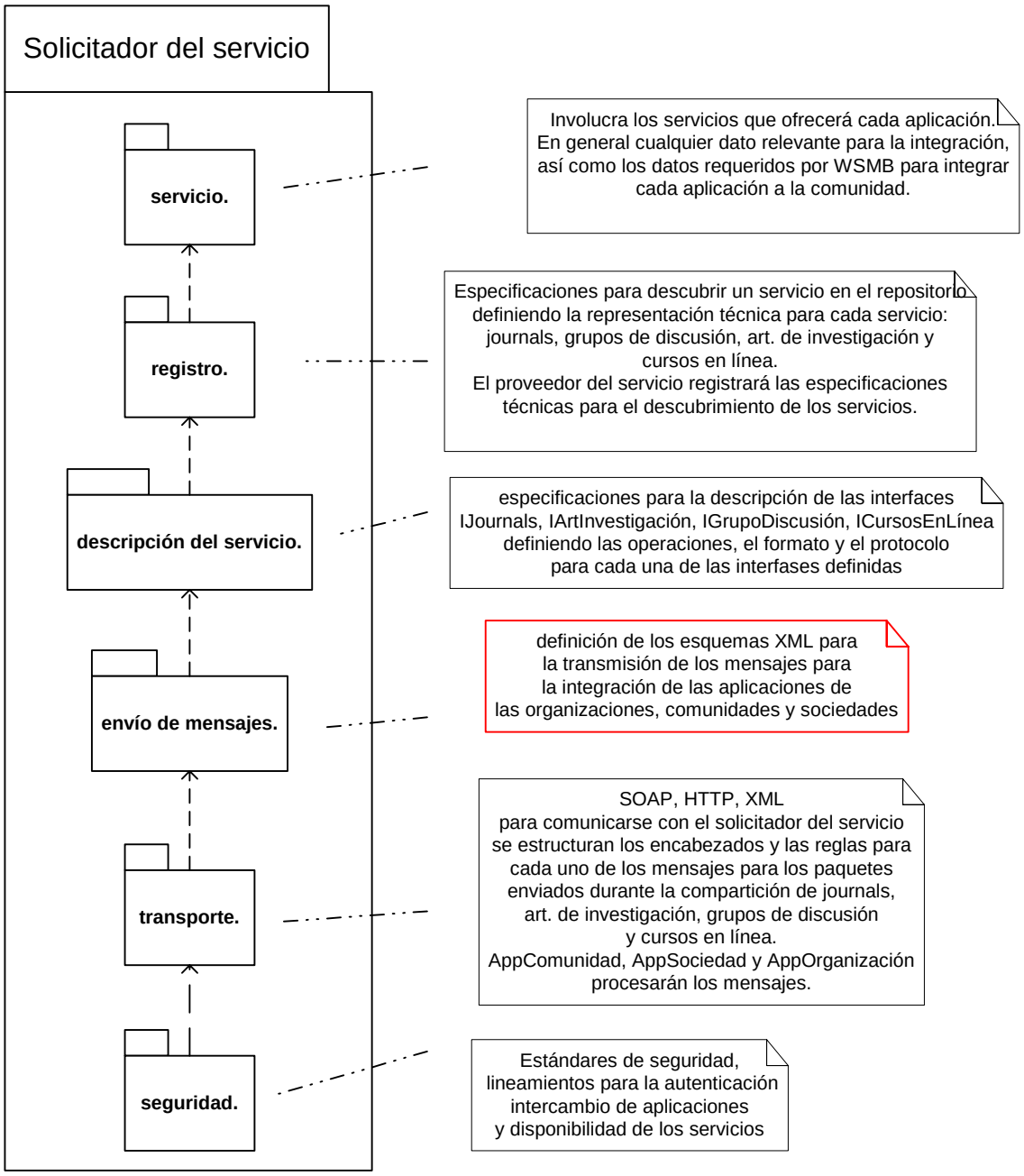


Figura C.19. Diagrama de la vista de pila para el proveedor del servicio

Cliente

En la figura C.20 muestra la vista de pila para el solicitador del servicio. Al igual que en la vista de pila para el proveedor, cada capa tiene comentados los aspectos que deben ser implementados.

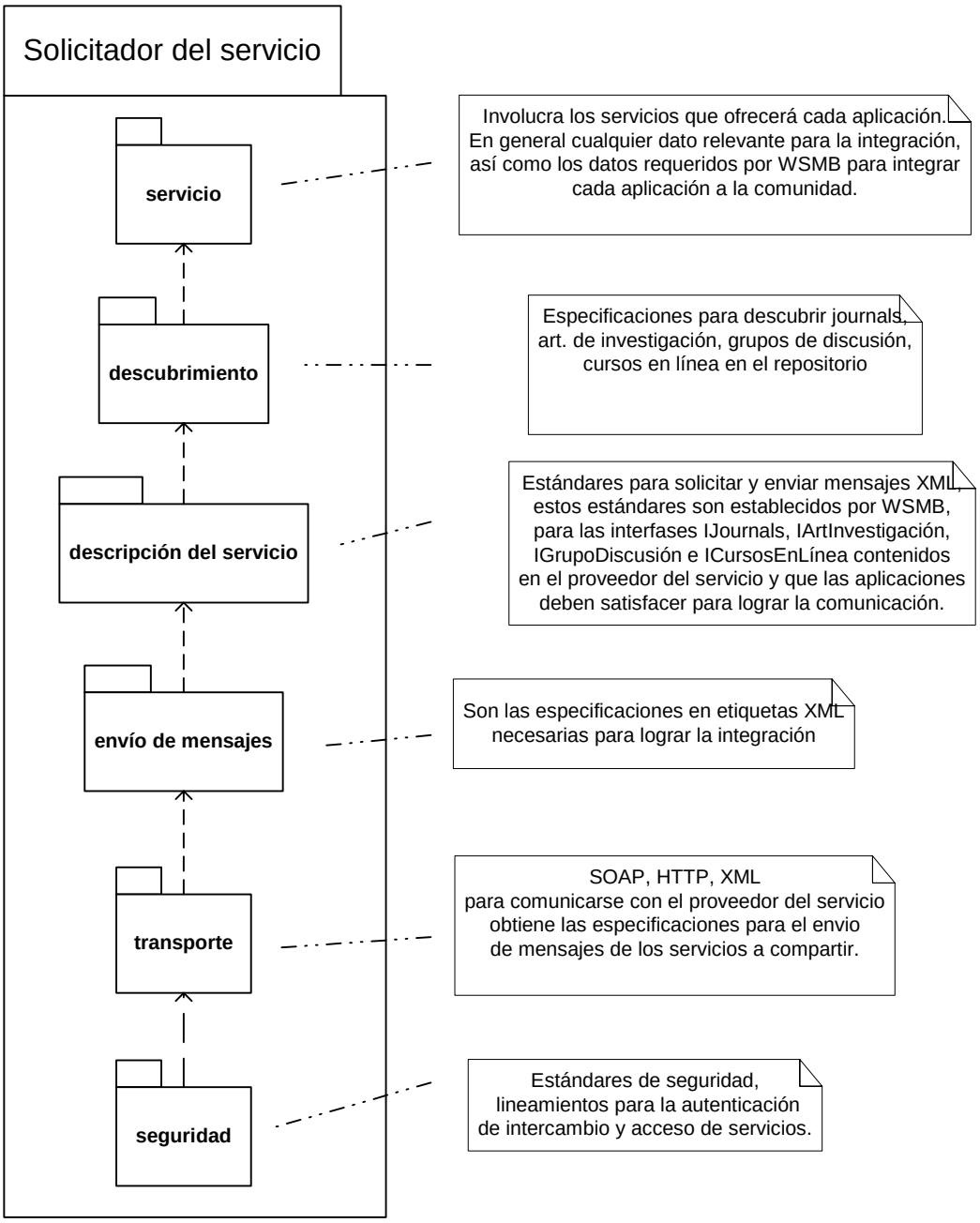


Figura C.20. Diagrama de la vista de pila para el solicitador del servicio

4.1.4.2. Vista lógica

La figura C.21 muestra la vista de servicios (vista lógica) para WSMB en la que un Web Services Cliente que utilice el portal y utilice los servicios lo hará como si se tratará de una sola interface, aunque el portal internamente tendrá

invocaciones hacia otros servicios. Se observa que este diagrama es una modelación diferente para el diagrama realizado durante la especificación de los requerimientos de integración. En otras palabras la modelación de los requerimientos de integración es útil para representar esta vista de la arquitectura.

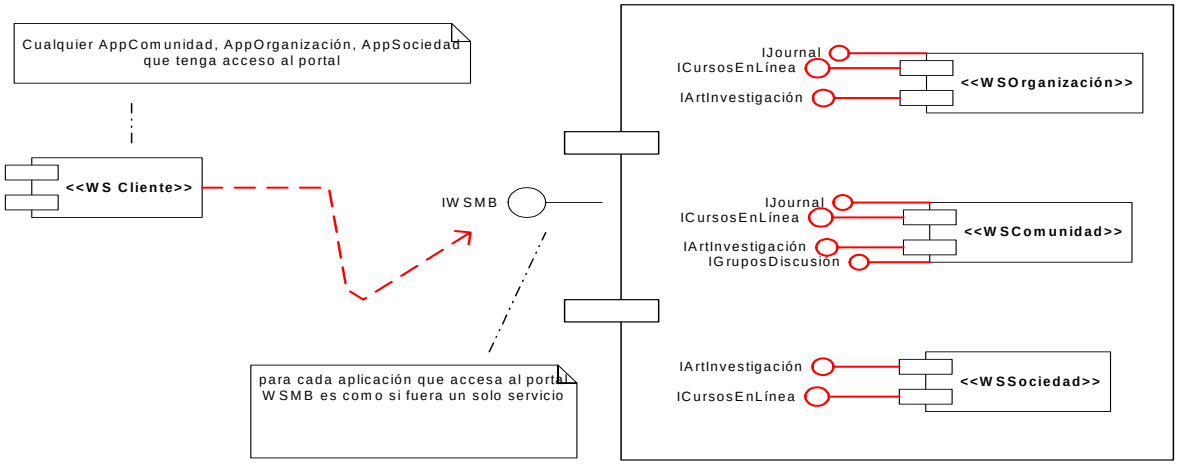


Figura C.21 Diagrama de la vista lógica (vista de servicios)

5. Análisis

5.1. Diagramas de clases

En el diagrama de clases se definen las clases que implementarán y utilizarán las interfaces y se identifican clases correspondientes a los estándares WSDL y UDDI. Se observa que en el diagrama C.22 no se han identificado atributos (solo la clase entidad tiene algunos atributos) y que a excepción de las relaciones estereotipadas para representar los estándares no se representan más relaciones entre clases. En esta figura se identifican la clase WSMB que realiza cuatro interfaces las cuales serán implementadas por las entidades, entre éstas clases se identifica la asociación <<SOAP>> la cual permite identificar el estándar que definirá la relación, se observan las clase registro relacionadas por la asociación estereotipada <<UDDI>> y la clase descripción que se relaciona por la asociación <<WSDL>>.

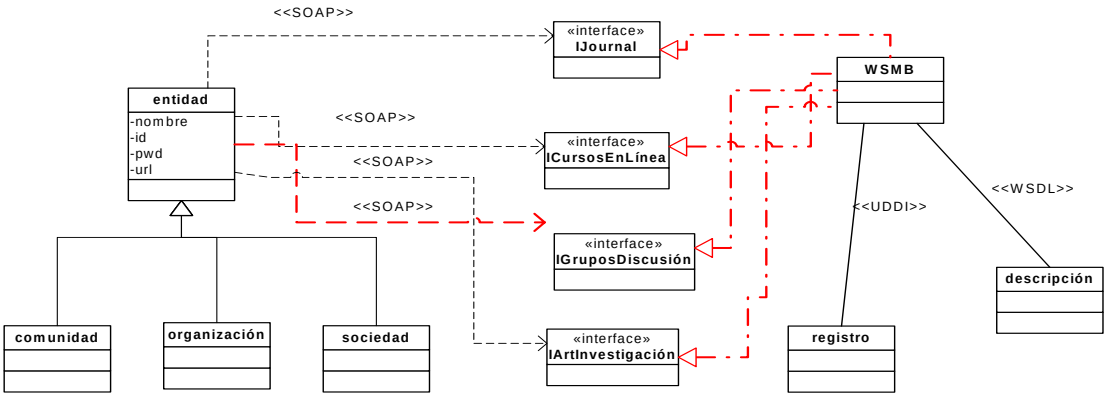


Figura C.22. Diagrama de clases en la fase del análisis

5.2. Diagramas de secuencia

En esta sección se muestran los diagramas de secuencia para los casos de uso definidos, tomando como base los diagramas del análisis de robustez. El objetivo de estos diagramas es mostrar la secuencia de mensajes que desencadenan una acción tal como se muestra en los siguientes diagramas.

Consulta cursos en línea

En el diagrama de secuencia mostrado en la figura C.23 se muestran los mensajes que inician este servicio. Los actores (AppComunidad, AppSociedad y AppOrganización) mandan mensajes para registrarse en el portal a un objeto interface llamado “cuadro de registro”; la clave y nombre de usuario son recibidos y verificados, si son correctos se envía un mensaje a la entidad “menú de servicio”, de lo contrario se regresa un mensaje de error. Si se mostró la lista de servicios, la entidad “menú de servicios” envía una solicitud para consultar el curso, si es correcto, éste objeto “selección de curso” envía un mensaje de acceso al curso y de mostrar servicio a la entidad “repositorio de los cursos”.

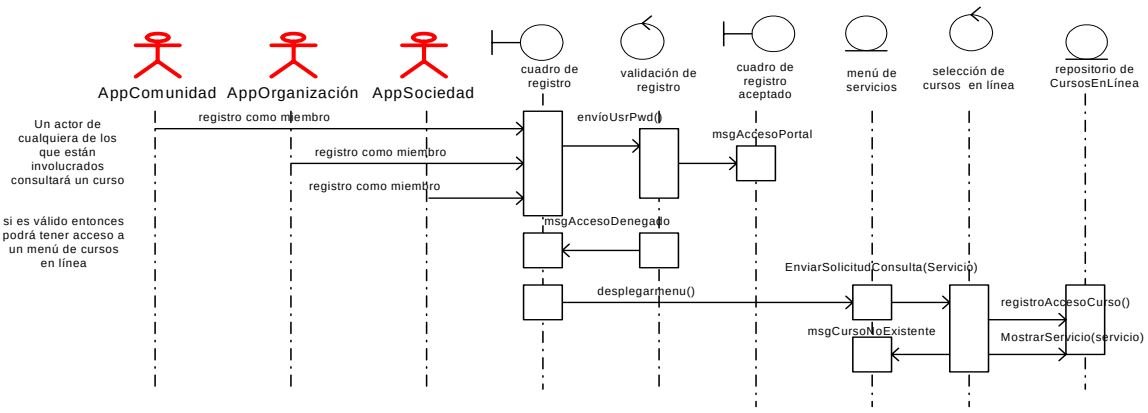


Figura C.23. Diagrama de secuencia para “consulta cursos en línea”

Consulta grupos de discusión

En el diagrama de secuencia mostrado en la figura C.24 se muestran los mensajes que inician este servicio. Los actores (AppComunidad, AppSociedad y AppOrganización) mandan mensajes para registrarse en el portal a un objeto interface llamado “cuadro de registro”; la clave y nombre de usuario son recibidos y verificados, si son correctos se envía un mensaje a la entidad “menú de servicio”, de lo contrario se regresa un mensaje de error. Si se mostró la lista de servicios, la entidad “menú de servicios” envía una solicitud para consultar el grupo de discusión, si es correcto, éste objeto “selección de grupo de discusión” envía un mensaje de acceso al grupo, opcionalmente podría participar en el foro, para lo cual se debe revisar el diagrama de secuencia “participa en grupos de discusión”.

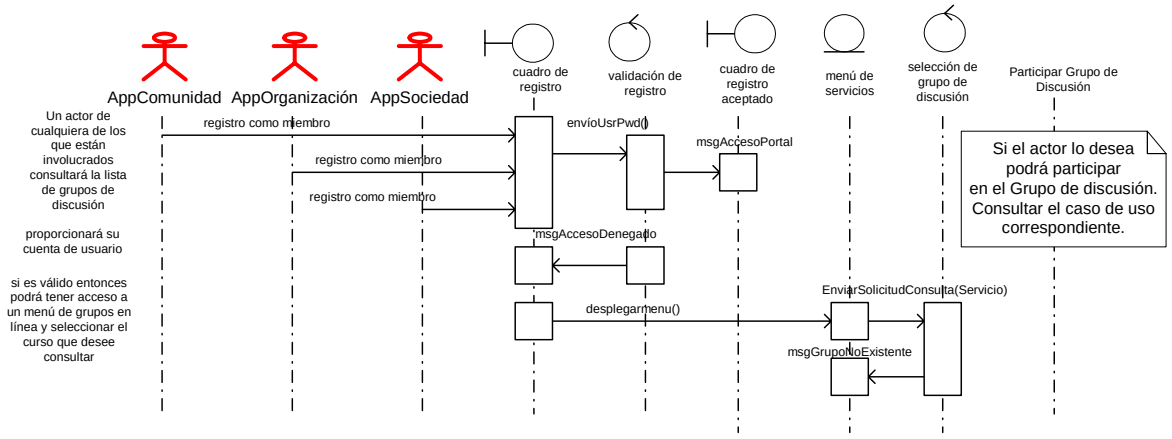


Figura C.24. Diagrama de secuencia para “consulta grupos de discusión”

Consulta *journals*

En el diagrama de secuencia mostrado en la figura C.25 se muestran los mensajes que inician este servicio. Los actores (AppComunidad, AppSociedad y AppOrganización) mandan mensajes para registrarse en el portal a un objeto interface llamado “cuadro de registro”; la clave y nombre de usuario son recibidos y verificados, si son correctos se envía un mensaje a la entidad “menú de servicio”, de lo contrario se regresa un mensaje de error. Si se mostró la lista de servicios, la entidad “menú de servicios” envía una solicitud para consultar el *journal*, si es correcto, éste objeto “selección de *Journal*” envía un mensaje a la entidad “repositorio de *journal”* para consultar el servicio.

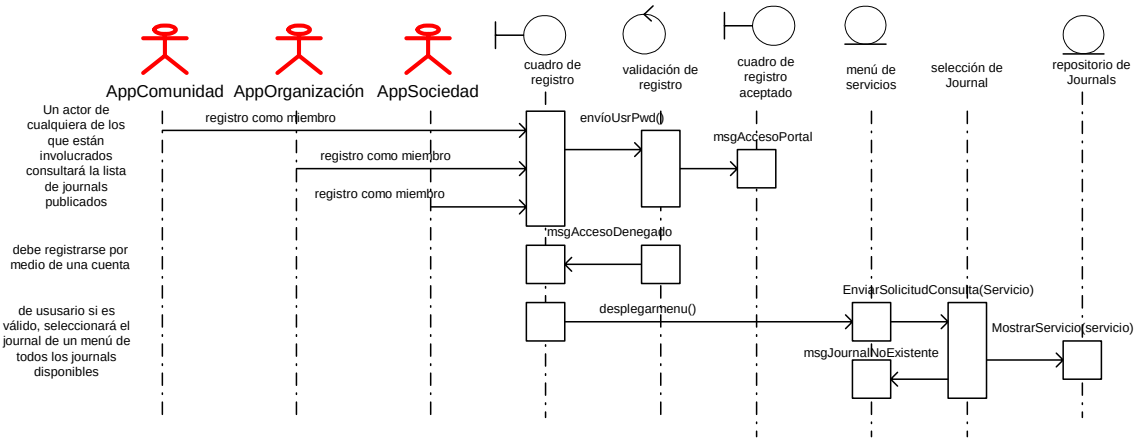


Figura C.25. Diagrama de secuencia para “consulta *journals*”

Consulta artículos de investigación

En el diagrama de secuencia mostrado en la figura C.26 se muestran los mensajes que inician este servicio. Los actores (AppComunidad, AppSociedad y AppOrganización) mandan mensajes para registrarse en el portal a un objeto interface llamado “cuadro de registro”; la clave y nombre de usuario son recibidos y verificados, si son correctos se envía un mensaje a la entidad “menú de servicio”, de lo contrario se regresa un mensaje de error. Si se mostró la lista de servicios, la entidad “menú de servicios” envía una solicitud para consultar el artículo de investigación, si es correcto, éste objeto “selección de artículo de investigación” envía un mensaje a la entidad “repositorio de ArtInvestigación” para consultar el servicio.

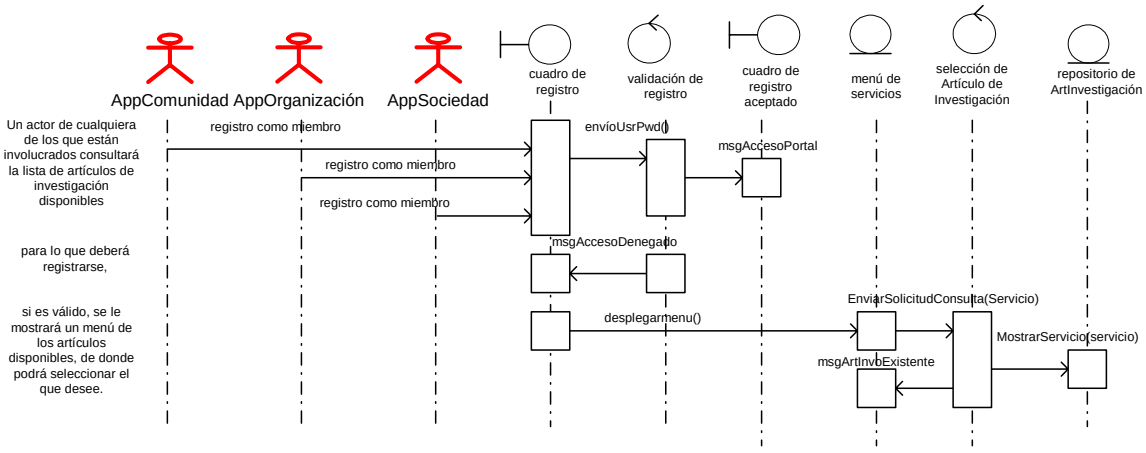


Figura C.26. Diagrama de secuencia para “consulta artículos de investigación”

Ofrece cursos en línea

La figura C.27 muestra el diagrama de secuencia cuando los actores (AppComunidad, AppSociedad y AppOrganización) ofrecen cursos en línea, éstos envían clave y nombre de usuario para registrarse en el portal a un objeto interface llamado “Solicitud de ingreso de Servicios”; si son correctos éste objeto envía un mensaje llamado “desplegarmenuServicio” a la entidad “repositorio de CursosEnLínea” con el curso que desea compartir. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

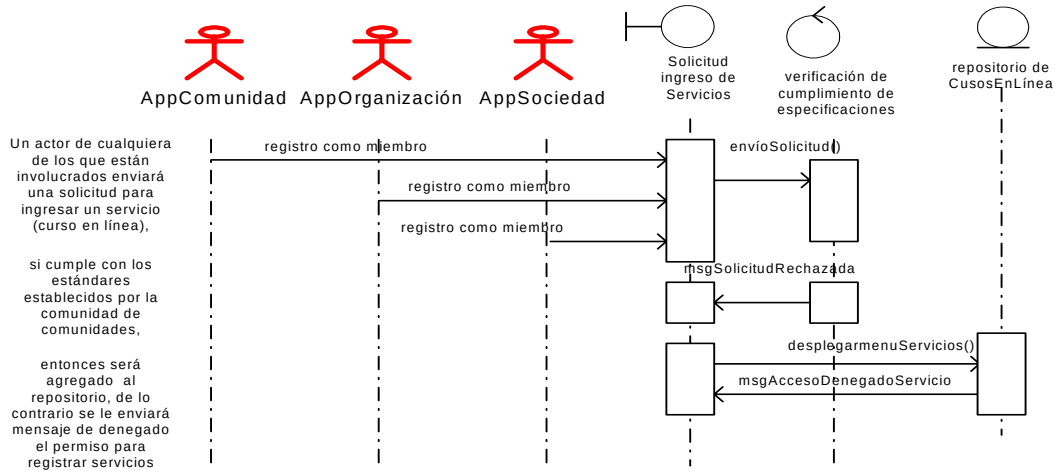


Figura C.27. Diagrama de secuencia para “ofrece cursos en línea”

Crea grupos de discusión

La figura C.28 muestra el diagrama de secuencia cuando el acto AppComunidad crea grupos de discusión, éste envía clave y nombre de usuario para registrarse en el portal a un objeto interface llamado “Solicitud de ingreso de Servicios”; si son correctos éste último objeto envía un mensaje llamado “crearGrupoDiscusión” a la entidad “Foro de discusión”. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

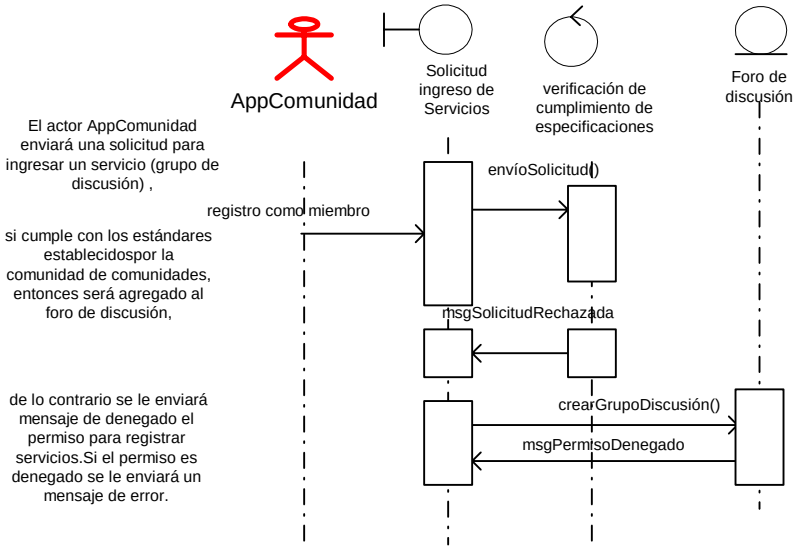


Figura C.28. Diagrama de secuencia para “ofrece cursos en línea”

Agrega *journals*

La figura C.29 muestra el diagrama de secuencia cuando los actores (AppComunidad y AppSociedad) agregan *journals*, éstos envían clave y nombre de usuario para registrarse en el portal a un objeto interface llamado “Solicitud de ingreso de Servicios”; si son correctos éste objeto envía un mensaje llamado “desplegarmenuServicios” a la entidad “repositorio de Journals”. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

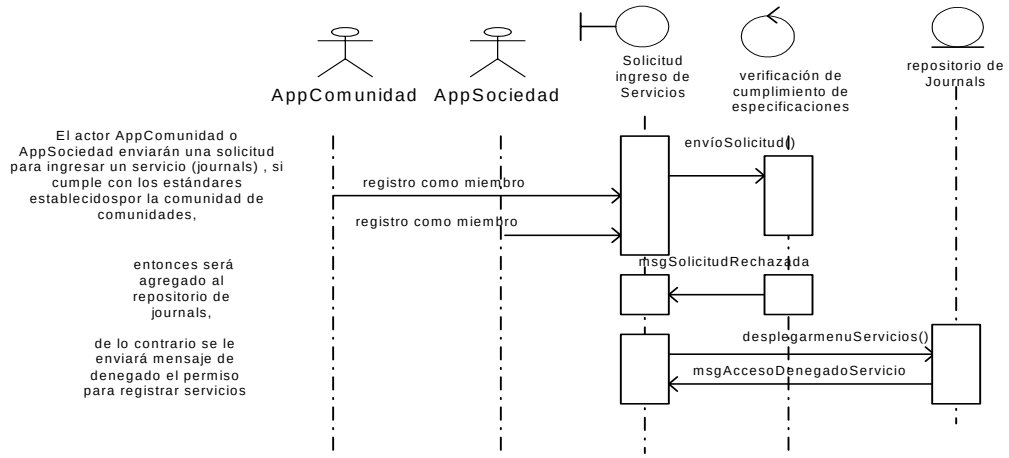


Figura C.29. Diagrama de secuencia para “ofrece cursos en línea”

Agrega artículos de investigación

La figura C.30 muestra el diagrama de secuencia cuando los actores (AppComunidad, AppOrganización y AppSociedad) agregan artículos de investigación, éstos envían clave y nombre de usuario para registrarse en el portal a un objeto interface llamado “Solicitud de ingreso de Servicios”; si son correctos éste objeto envía un mensaje llamado “desplegarmenuServicios” a la entidad “repositorio de ArtInvestigación”. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

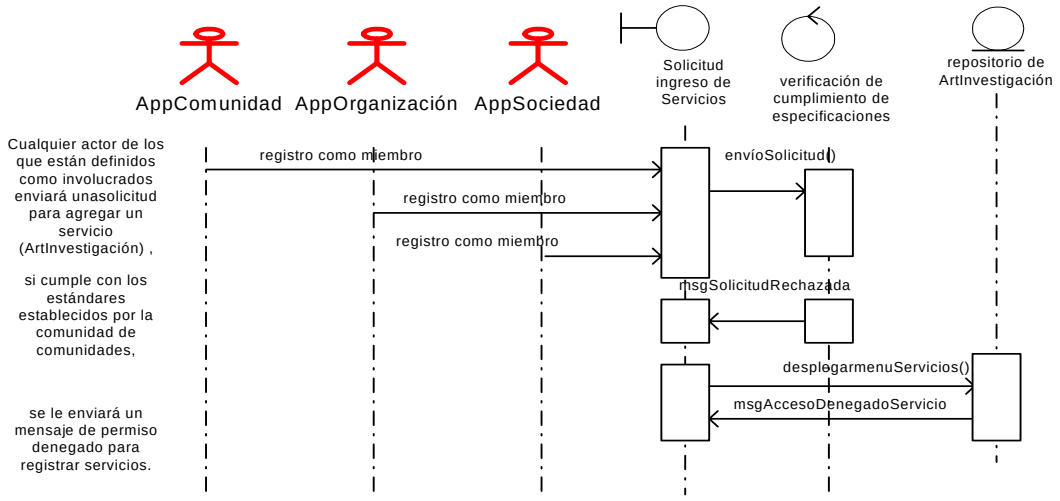


Figura C.30. Diagrama de secuencia para “agrega artículos de investigación”

Participa en grupos de discusión

La figura C.31 muestra el diagrama de secuencia cuando los actores (AppComunidad, AppOrganización y AppSociedad) participan en grupos de discusión, éstos envían clave y nombre de usuario a un objeto interface llamado “cuadro de registro”; si son correctos los datos éste objeto envía un mensaje llamado “desplegarmenu” a la entidad “menú de servicios”, el cual envía un mensaje “RegistrarEnGpo” a la entidad foro de discusión. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

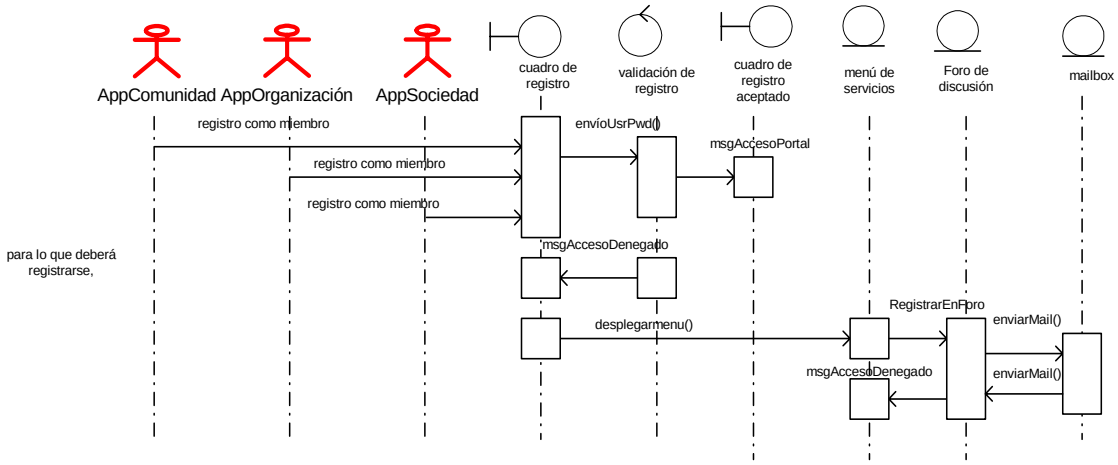


Figura C.31. Diagrama de secuencia para “participa en grupos de discusión”

Obtiene *journals* y artículos de investigación

La figura C.32 muestra el diagrama de secuencia cuando los actores (AppComunidad, AppOrganización y AppSociedad) quieren obtener *journals* o artículos de investigación participan en grupos de discusión, éstos envían clave y nombre de usuario a un objeto interface llamado “cuadro de registro”; si son correctos los datos éste objeto envía un mensaje llamado “desplegarmenu” a la entidad “Repositorio de ArtInvestigación” y otro mensaje “desplegarmenu” a la entidad llamada “Repositorio de Journals”. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

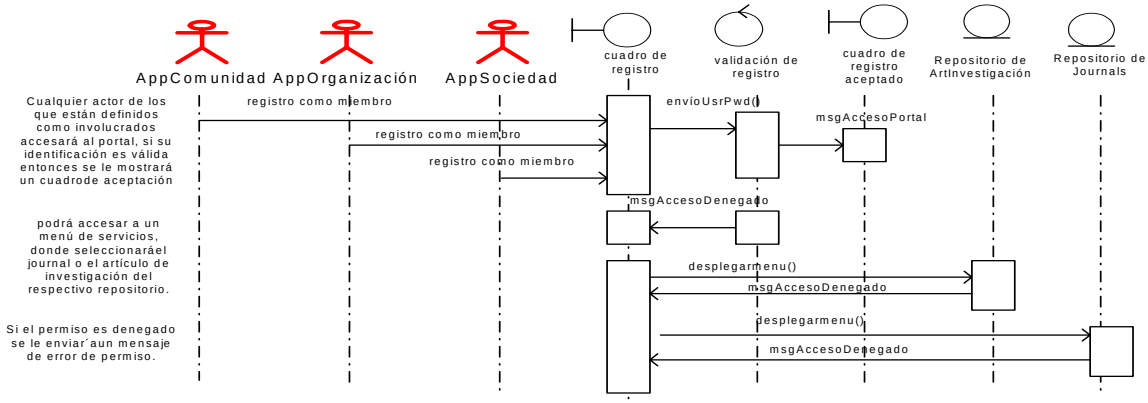


Figura C.32. Diagrama de secuencia para “obtiene *journals* y artículos de investigación”

Inscripción a cursos en línea

La figura C.33 muestra el diagrama de secuencia cuando los actores (AppComunidad, AppOrganización y AppSociedad) quieren inscribirse a algún curso en línea, éstos envían clave y nombre de usuario a un objeto interface llamado “cuadro de registro”; si son correctos los datos éste objeto envía un mensaje llamado “desplegarmenu” a la entidad “menú de servicios”, el controlador “validación de CursoEnLínea” evalúa los datos y envía mensaje “agregarMiembroCursoE” a la entidad “repositorio CusosEnLínea”. Si el repositorio rechaza este mensaje debe responderle al objeto “Solicitud ingreso de servicio” con un mensaje de notificación de rechazo.

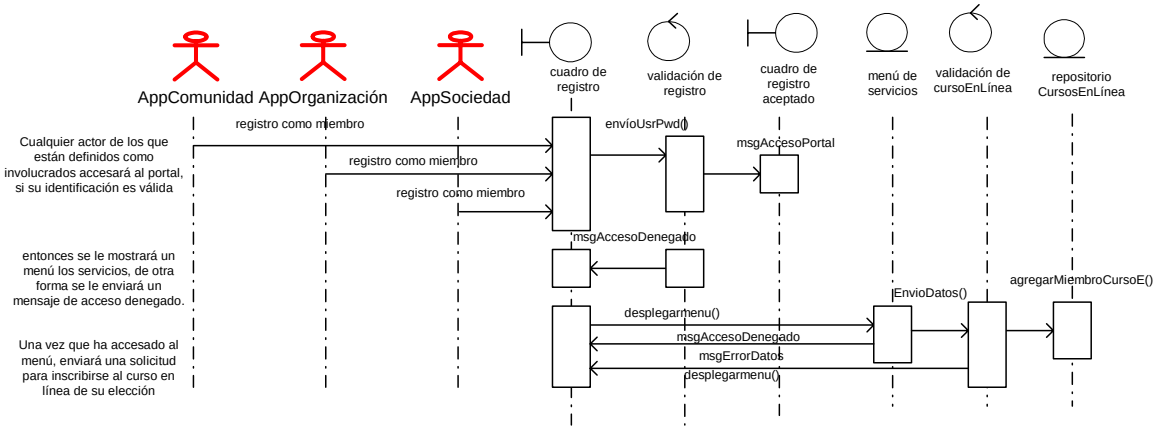


Figura C.33. Diagrama de secuencia para “inscripción a cursos en línea”

Alta de aplicación

La figura C.34 muestra el diagrama de secuencia cuando el administrador del portal registrará a una aplicación en el portal, esto inicia por medio de un mensaje en el que el actor envía la solicitud a un objeto interfase llamado “cuadro de administración” si la solicitud es válida éste último objeto, transfiere esos datos al objeto interfase “cuadro de registros aceptado”. El controlador “verificación de usuario y clave” valida los datos y si son correctos, le envía el mensaje “agregaRegistro” a la entidad “BDAplicaciones”.

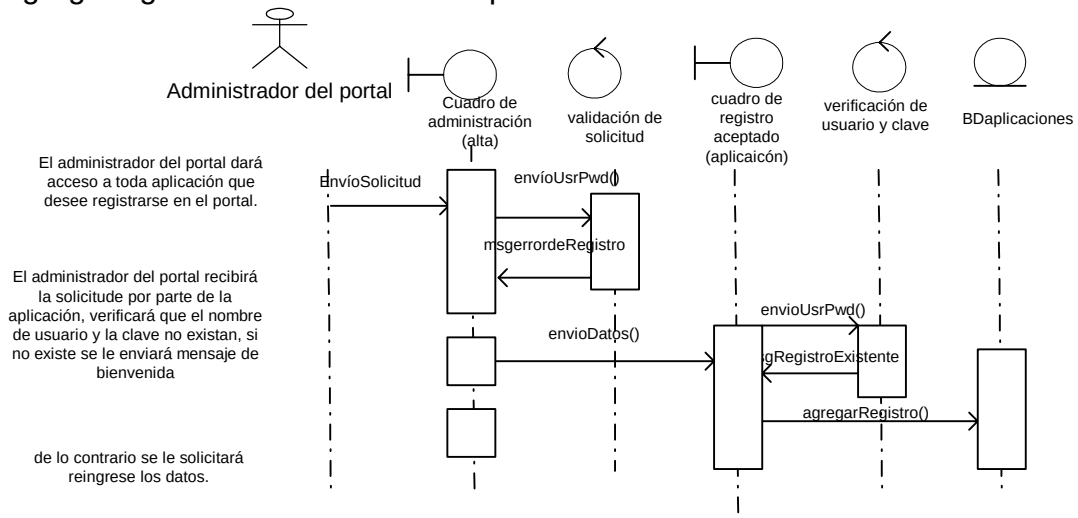


Figura C.34. Diagrama de secuencia para “alta de aplicación”

Baja de una aplicación al portal

La figura C.35 muestra el diagrama de secuencia cuando el administrador del portal eliminará del registro del portal a una aplicación, esto inicia por medio de un mensaje en el que el actor envía la solicitud a un objeto interfase llamado “cuadro de administración” si la solicitud es válida éste último objeto, transfiere esos datos al controlador “verificación de usuario y clave”. Éste controlador envía un mensaje de “EliminarRegistro” a la entidad “BDAplicaciones”.

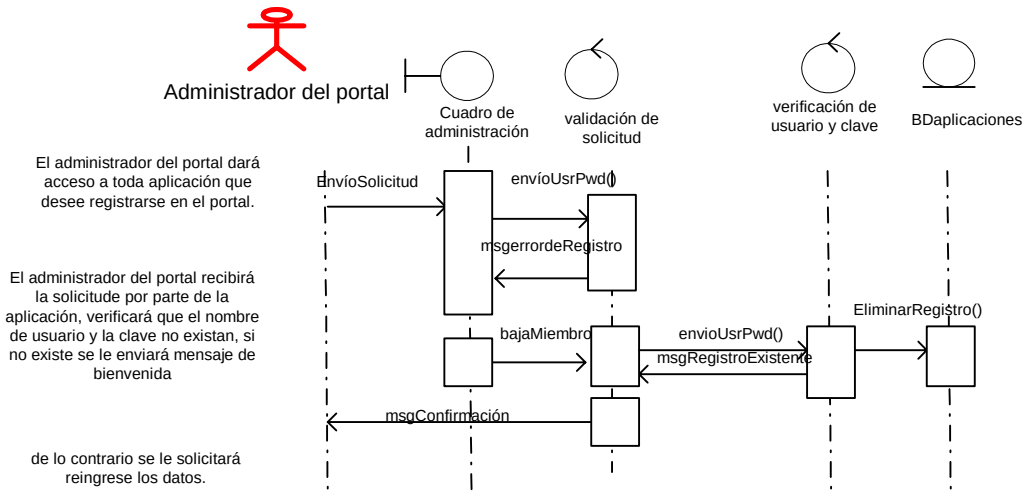


Figura C.35. Diagrama de secuencia para “baja de aplicación”

Actualiza datos

La figura C.36 muestra el diagrama de secuencia cuando el administrador del portal desea actualizar datos de las aplicaciones, esto inicia por medio de un mensaje en el que el actor envía la solicitud a un objeto interfase llamado “cuadro de administración” si la solicitud es válida éste último objeto, transfiere esos datos al controlador “verificación de usuario y clave”. Éste controlador envía un mensaje de “ModificarRegistro” a la entidad “BDAplicaciones”.

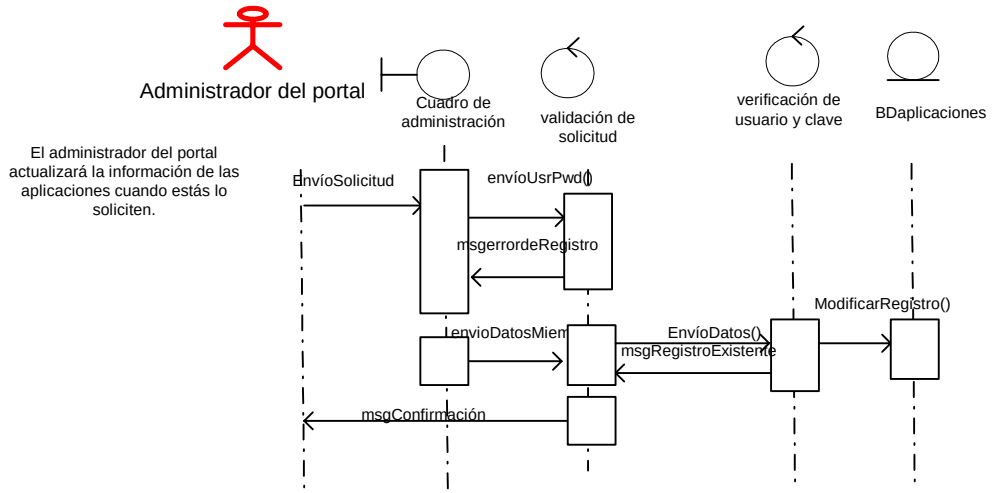


Figura C.36. Diagrama de secuencia para “baja de aplicación”

5.3 Diagramas de actividad

En la identificación de los casos de uso estos fueron divididos en cuatro grupos: búsqueda de servicios, publicación de servicios, uso de servicios y registro de aplicaciones.

En la figura C.37 se muestra el diagrama de actividad correspondiente a la búsqueda de servicios, en esta figura se identifican las actividades necesarias por parte de los clientes (AppComunidad, AppSociedad, AppOrganización) y por parte del proveedor WSMB para la realización de la búsqueda de servicios. Los clientes inician esta búsqueda y el WSMB a través de las interfaces deberá obtener las descripciones para poder satisfacer la solicitud del servicio y que el cliente pueda satisfacer su petición.

Búsqueda de servicios

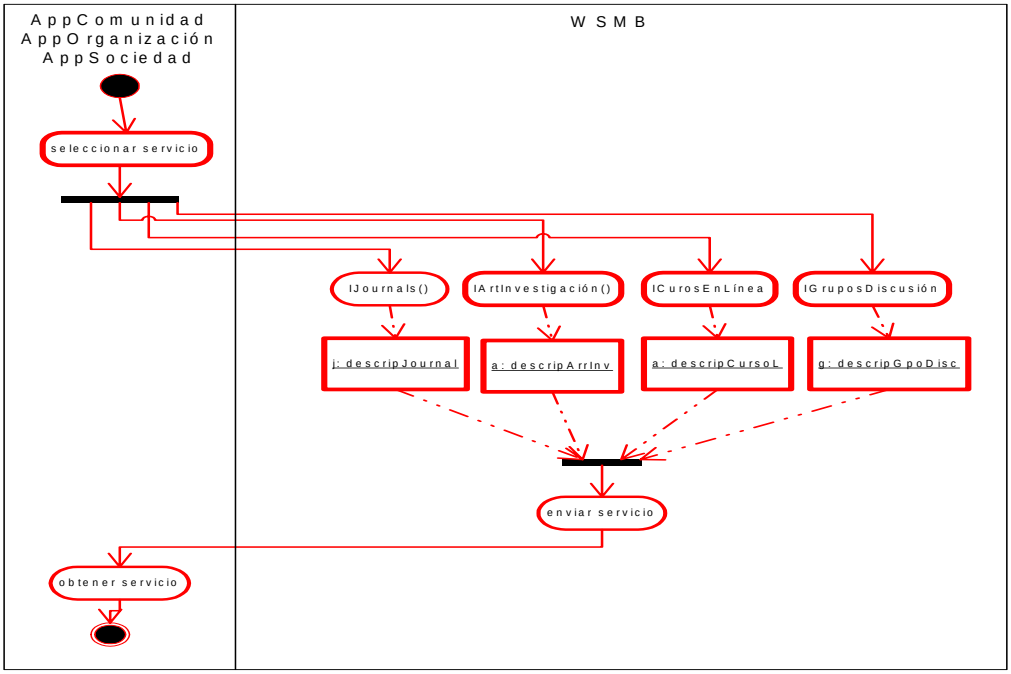


Figura C.37. Diagrama de actividad para la búsqueda de servicios

Publicación de servicios

En la figura C.38 se muestra el diagrama de actividad correspondiente a la publicación de servicios, en esta figura se identifican las actividades necesarias por parte de los clientes (AppComunidad, AppSociedad, AppOrganización) y por parte

del proveedor WSMB para el registro de una descripción. Los clientes generan la descripción de cualquiera de los servicios por medio de las descripciones para cada servicio. WSMB verifica que la descripción sea correcta y registra el servicio con la descripción proporcionada.

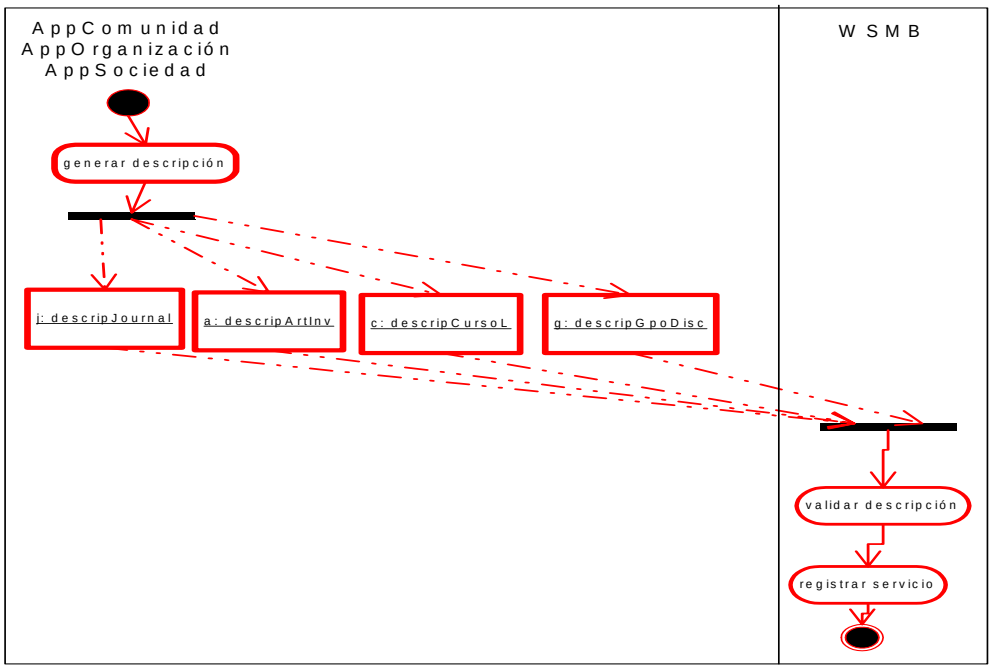


Figura C.38. Diagrama de actividad para la publicación de servicios

Uso de servicios

En la figura C.39 se muestra el diagrama de actividad correspondiente al uso de los servicios, en esta figura se identifican que los clientes (AppComunidad, AppSociedad, AppOrganización) seleccionan el servicio al proveedor (WSMB) el cual proporcionará la descripción del servicio para que este pueda ser utilizado por los clientes a través de las interfaces y puedan obtener el servicio solicitado.

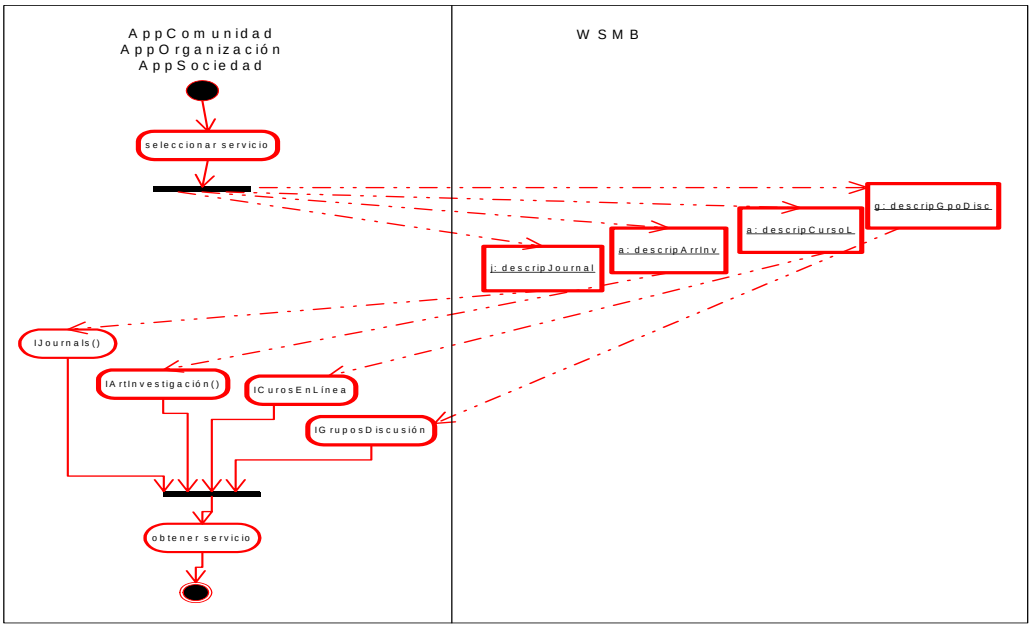


Figura C.39. Diagrama de actividad para el uso de servicios

Registro de aplicaciones

En la figura C.40 se muestra el diagrama de actividad correspondiente al registro de aplicaciones. En esta figura se muestra que el administrador del portal obtendrá la solicitud para realizar un registro, una baja o actualizar datos de las aplicaciones en la comunidad. WSMB verificará que estos datos sean correctos y actualizará la base de datos de aplicaciones.

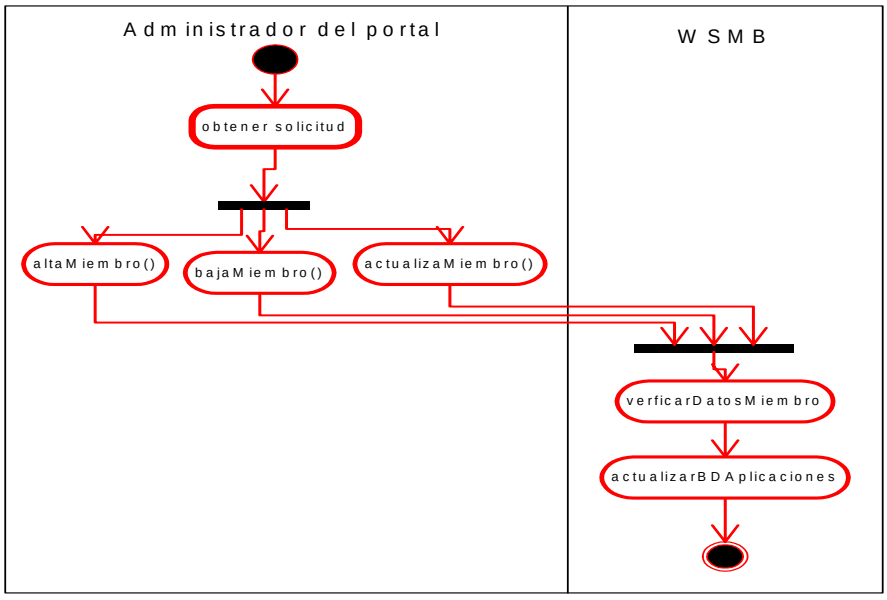


Figura C.40. Diagrama de actividad para el registro de aplicaciones

6. Diseño

6.1. Diagramas de clases

La figura C.41 muestra el diagrama de clases elaborado en la fase del análisis, pero en esta fase se definen las operaciones que contendrá cada interface que finalmente son los métodos a los que los clientes podrán accesar y mediante los cuales WSMB integrará las aplicaciones y permitirá la compartición de servicios. Se identifican también las operaciones de las clases registro y descripción.

Es importante aclarar que para fines ilustrativos en este diagrama se han omitido los atributos, sin embargo para esta fase ya deben estar definidos.

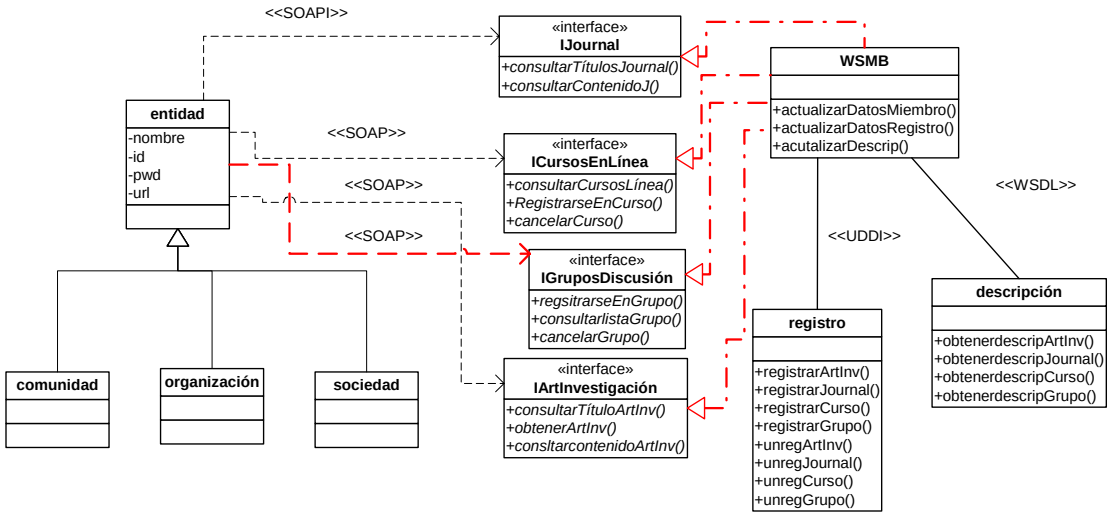


Figura C.41. Diagrama de clases elaborado en la fase de diseño

6.2. Diagramas de colaboración

En la identificación de los casos de uso estos fueron divididos en cuatro grupos: búsqueda de servicios, publicación de servicios, uso de servicios y registro de aplicaciones.

Búsqueda de servicio

En la figura C.42 se muestra el diagrama de colaboración correspondiente a la búsqueda de servicios. Inicialmente se solicita el servicio, se obtendrá la

solicitud que podría corresponder a cualquiera de las interfaces que implementen los servicios, una vez que se tiene la solicitud del servicio buscado se obtiene la descripción que es enviada al repositorio para que sea satisfecha esta solicitud y regrese al objeto que inició la interacción.

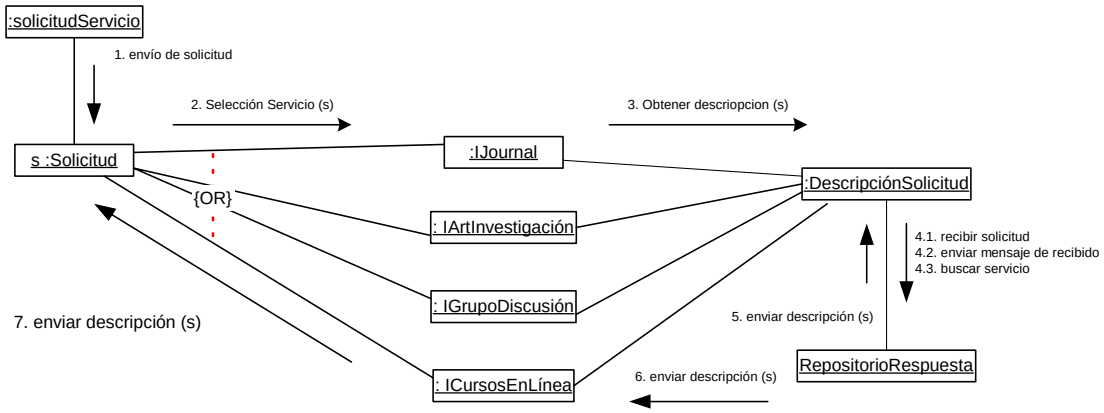


Figura C.42. Diagrama de colaboración para la búsqueda de servicio

Publicación de servicios

En la figura C.43 se muestra el diagrama de colaboración correspondiente al uso de servicios. Inicialmente se genera la descripción, la cual es enviada a los diferentes servicios para obtener la descripción de la solicitud. Se verifica que la descripción cumpla con las especificaciones. Si es correcta, el repositorio registra el servicio y notifica por medio de un mensaje.

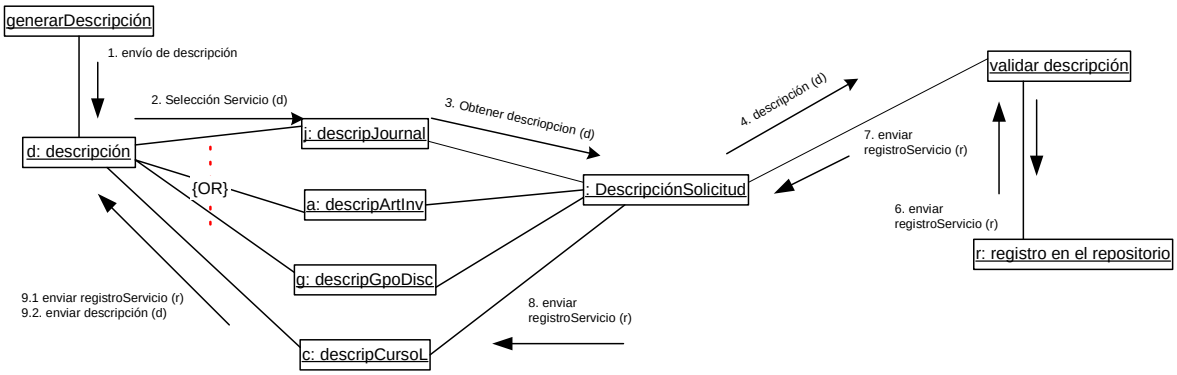


Figura C.43. Diagrama de colaboración de la búsqueda de servicio

Uso de servicios

En la figura C.44 se muestra el diagrama de colaboración correspondiente a la publicación de servicios. Inicialmente se genera la descripción, la cual es enviada a los diferentes servicios para obtener la descripción del servicio solicitado. Si la descripción es correcta se selecciona el servicio del registro de repositorio, quien enviará el servicio solicitado para que pueda ser utilizado.

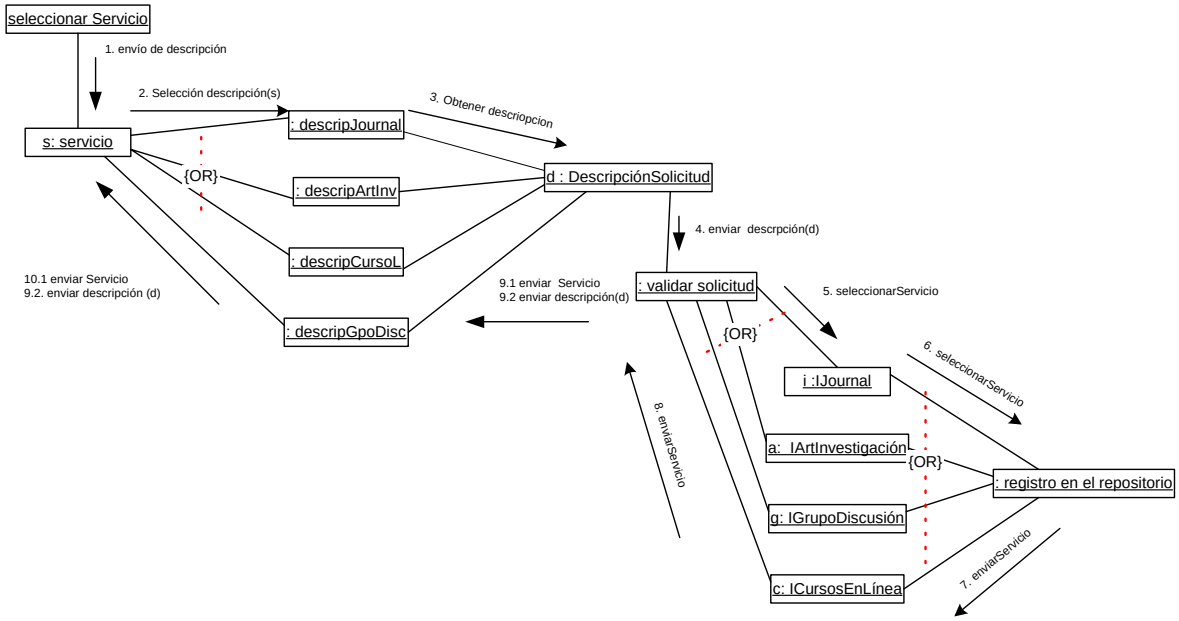


Figura C.44. Diagrama de colaboración de la búsqueda de servicio

Registro de aplicaciones

En la figura C.45 se muestra el diagrama de colaboración correspondiente al registro de aplicaciones. Inicialmente se obtiene la solicitud para realizar una alta, baja o actualizar datos de los miembros. Se envían los datos obtenidos en esta solicitud para que sean verificados. Si los datos están correctos se actualiza en la base de datos, dependiendo de la operación que se haya solicitado.

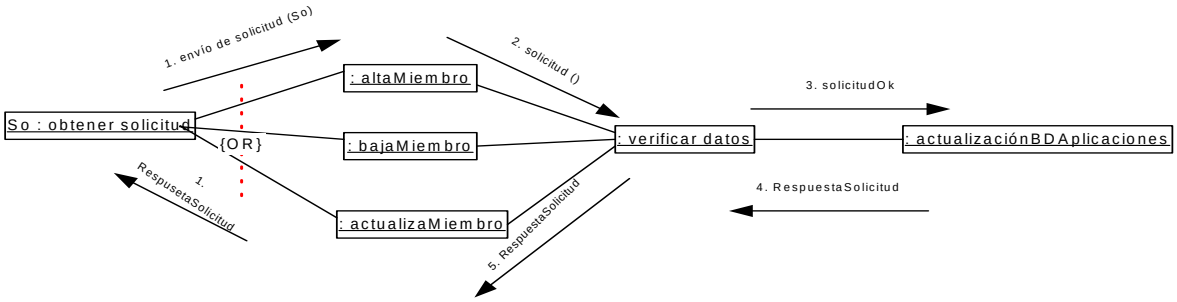


Figura C.45. Diagrama de colaboración del registro de aplicaciones

7. Codificación e Implementación y Pruebas

Estas fases han sido omitidas debido a que no se llegó a implementar físicamente dicho sistema. Aún cuando este sistema ya está implementado; en este trabajo se hizo la modelación con el proceso propuesto en el capítulo 3.

8. Resumen

En este anexo se presentó el documento completo de la modelación realizada a Web Services MedBiquitous. Se omitieron algunos aspectos como el diccionario de datos porque se considera un documento técnico y el objetivo fue validar si los diagramas propuestos eran suficientes para la modelación del caso de estudio, también se omitieron las fases de codificación e implementación y pruebas, debido a que en este trabajo no se llegó a la implementación de ninguno de ambos casos de estudio.

El resultado fue que los diagramas resultaron convenientes para representar cada fase del proceso. A excepción del diagrama de clases para representar la arquitectura el cual no pudo ser aplicado ya que se carecían de especificaciones suficientes para poder ser aplicado

Referencias

[AFCJ2001] Akkiraju R, Flaxer D, Chang H, Chao T, Zhang L and Jeng J, “A Framework for Facilitation Dynamic e-Business Via Web Services”, Comm. ACM, Aug 2001.

[Arm2002] Armstrong Chris, www.omg.org/…/workshops/presentations/Web Services_2002/03-2_Armstrong-odelingWebService/s_with_UML.pdf, página visitada: 6/may/2003

[BO2001] Baker, S. and O'Sullivan, D. “Positioning CORBA, J2EE, web services and other Middlewares”, Distributed Objects and Applications, 2001. DOA '01. Proceedings. 3rd International Symposium on, 2001, Page(s): 359 –360.

[BRJ1999] Booch G., Rumbaugh J., Jacobson I.; “The Unified Modeling Language User Guide”, Addison-Wesley, 1999.

[Can2003] Canché, L., “Implementación de Web Services en un sistema MES”, propuesta de tesis de maestría.

[Car2001] Carlson, D., xmlmodeling.com/examples/uddi/ModelingUDDI.pdf, página visitada: 20/ago/2002

[Cla2002] Clark, D., “Next-generation web services”, IEEE Internet Computing , Volume: 6 Issue: 2 , March-April 2002, Page(s): 12 -14

[Con2000] Conallen, J.; “Building Web Applications with UML”, Addison-Wesley, 2000.

[Con1999] Conallen, J., “Modeling web application architectures with UML”, Comm. ACM, 42(10), 1999.

[Gar2003] García, A., “Modelo cooperativo para la administración del contenido de cursos en línea. Un enfoque basado en Web Services”, propuesta de tesis de maestría.

[KJKS12001] Kirda, E., Jazayeri, M., Kerer, C. and Schranz, M., “Experiences in engineering flexible Web services”, IEEE Multimedia, Volume: 8 Issue: 1, Jan.-March 2001, Page(s): 58 –65.

[Kre2001] Kreger, H., “Web Services Conceptual Architecture (WSCA 1.0)”, IBM Software Group, May 2001, http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, página visitada: 3/ene/2003

[Kru2000] Kruchten, P.; “The Rational Unified Process an Introduction Second Edition”, Addison-Wesley, 2000.

[GSBDDNN2002] Graham, S., Simeonov, S., Boubez, T., Davis, D., Daniels, G., Nakamura, Y., and Neyama, R., “Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI.”, Sams, 2002.

[Lar1999] Larman, C.; “UML and Patterns, introduction to Object-Oriented Analysis and Design”; Prentice-Hall, 1999.

[LF2002] Lublinsky, B., Farrel M.; “Web Services The implementation Iceberg”, eAIJournal, 2002, www.eaijournal.com/PDF/WebServicesLublinsky.pdf, página visitada: 3/mar/2002

[Mc2001] McNay, “UML for e-business: New Use for Use Cases”, Professional Communication Conference, 2001. IPCC 2001. Proceedings. IEEE International, 2001, Page(s): 245 -249

[Mye2001] Myerson, J., “Web Services Architectures”, 2001, www.webservicesarchitect.com/content/articles/webservicesarchitectures.pdf, página visitada: 3/ene/2003

[NHN2002] Nakamur Y., Hada S., Neyama R.,”Towards the integration of Web services security on enterprise environments”,Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on, 2002, Page(s): 166 - 175

[Oell2001] Oellermann, W.; “Architecting Web Services”; Apress, 2001.

[Oes1999] Oestereich, B.; “Developing Software with UML”, Addison-Wesley, 1999.

[OMG2001] OMG Unified Modeling Languaje Specification v 1.4, Sep 2001.

[RS1999] Rosenberg, D. with Scott, K.; “Use Case Driven Object Modeling with UML, a practical approach”, Addison-Wesley, 1999.

[RR2001] Roy, J. and Ramanujan, A., “Understanding web services”, IT Professional, Volume: 3 Issue: 6 , Nov/Dec 2001, Page(s): 69 –73.

[Rum1991] Rumbaugh, J., et al, “Object-Oriented Modeling and Design”, Englewood Cliffs, NJ., Prentice Hall, 1991.

[Vau2002] Vaughan-Nichols, S.J., “Web services: beyond the hype”, Computer, Volume: 35 Issue: 2, Feb 2002, Page(s): 18 –21.

[Vin2002] Vinoski, S., “Web services interaction models. I. Current practice”, IEEE Internet Computing, Volume: 6 Issue:3, May/Jun 2002, Page(s): 89 –91