Lógica de Mediación para la Interacción de Servicios basados en SOA e IMS

Ximena Velasco Melo

Tutor: Mag. Oscar Mauricio Caicedo Rendón

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telemática

Línea de Investigación en Servicios Avanzados de Telecomunicaciones

Popayán, Junio de 2011

Tabla de contenido

Tabla de Figuras

Lista de Tablas

Capítulo 1

INTRODUCCIÓN

1.1. Definición del Problema

Cuando los servicios de telecomunicaciones están atados a las tecnologías de la red subyacente se tiene la limitación de que éstos sean dependientes y que a su vez evolucionen lentamente, respecto a la exigencia que introduce la velocidad de cambio sin precedentes de las tecnologías utilizadas para el aprovisionamiento de servicios en la Internet. En este sentido, es necesario que las redes de telecomunicaciones cuenten con arquitecturas de prestación de servicios que permitan la habilitación y entrega de nuevos servicios, mejorándolos y haciéndolos evolucionar, a un ritmo similar al impuesto por los drásticos cambios de las TIC (Tecnologías de la Información y las Comunicaciones). Además, se requiere de mecanismos para que los servicios se creen de manera rápida y puedan ser soportados por cualquier red (existente o futura). En esta vía, se debe tener en cuenta que la mayoría de las empresas han hecho grandes inversiones en los recursos de las infraestructuras de acceso, transporte, control y servicios, por lo que la mejor solución no es descartar los sistemas existentes, sino hacer que evolucionen y mejoren. Lo anterior indica entonces que para los operadores de los sistemas de telecomunicaciones actuales y futuros, es fundamental la rentable, rápida, flexible y fácil provisión de servicios, y para la protección de las inversiones existentes es crucial la habilidad de integrar tanto las nuevas tecnologías como las anteriores (AePONA, 2005).

Al comparar los Servicios de Telecomunicaciones con los basados en Internet, se observa que para los primeros aún es desafiante el ciclo de desarrollo en términos de tiempo y costo debido principalmente a la manipulación directa de protocolos muy especializados; en el caso de los segundos, este problema se ha solventado con el uso de mediadores típicamente representados por librerías en un lenguaje de programación determinado. Entre las tecnologías, estándares y librerías adoptadas o en proceso de adopción dentro de los entornos de Telecomunicaciones y de las TIC se encuentran el Protocolo de Inicio de Sesión (Session Initiation Protocol, SIP) para los procesos de señalización, el Proyecto Conjunto de Tercera Generación (Thrid Generation Partnership Project, 3GPP), la Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA), los Servicios Web (Web Service, WS) para entornos distribuidos, y la Próxima Generación de Redes (Next Generation Network, NGN), término que mejor define la situación actual del entorno de las telecomunicaciones; adicionalmente, como una propuesta basada en NGN está el Subsistema Multimedia IP (IP

Multimedia Subsystem, IMS), una tecnología que representa los nuevos retos en convergencia e interoperabilidad de las próximas redes (Expocomm Argentina, 2005).

Por otro lado, para reducir el tiempo de salida al mercado de los servicios de telecomunicaciones y adicionarles capacidades propias de la red de redes, en los últimos años la convergencia de estos dos mundos (servicios de telecomunicaciones y servicios basados en Internet) ha adquirido un interés considerable (Tarkoma, y otros, 2008). En este sentido, es bien conocido que al utilizar un enfoque SOA para el desarrollo de aplicaciones compuestas e interoperables en el entorno de las Tecnologías de la Información (Information Technologies, IT), se hace posible la combinación de servicios de múltiples organizaciones y dominios de aplicación, utilizando interfaces basadas en estándares (Telefónica, 2004). Así, a la hora de dar soluciones eficaces a los problemas y retos del actual entorno de las telecomunicaciones, entre los que se encuentran el diseño y el despliegue de servicios convergentes y de valor agregado en las NGN y específicamente en IMS, se puede utilizar SOA, y en particular los WS que son la forma más difundida para su implementación (Baravaglio, y otros, 2005).

A pesar de lo expuesto, actualmente existen pocos mecanismos, y los existentes no están estandarizados, orientados a generar la interoperabilidad entre los WS (tecnología de la información) y el protocolo SIP de IMS (tecnología de las telecomunicaciones) (Zielinski, 2007). En este nuevo contexto tecnológico, donde es preponderante la creación rápida de servicios, es más eficiente la estimulación del desarrollo de servicios de próxima generación con y por parte de expertos en IT, ya que generalmente en ese ambiente las aplicaciones se desarrollan a alto nivel, utilizando mediadores, y no a bajo nivel, utilizando directamente los protocolos de comunicación. De esta forma, los sistemas heredados e IMS podrán seguir proporcionando las capacidades de la red subyacente, y el desarrollador no necesitará tener conocimiento profundo de los protocolos específicos de las capas inferiores, lo que finalmente le permite concentrarse en el acceso simple y de alto nivel a una capacidad de telecomunicación, utilizando técnicas comunes de programación, típicamente basadas en WS. En este sentido, los proveedores de servicios necesitarán entender las posibles interacciones entre el mundo de las IT, con SOA y los WS, y el mundo de las telecomunicaciones, con SIP de IMS, para generar los mecanismos que impulsen este nuevo entorno de convergencia a través del desarrollo y despliegue rápido de servicios compatibles con ambos mundos (Schwartz, 2006) (Light Reading, 2006).

A partir de la necesidad de proporcionarle a los proveedores de servicios una facilidad software que combine el mundo SIP/IMS de las Telecomunicaciones y el mundo SOA de las IT, para el desarrollo, despliegue y la prestación rápida y eficiente de servicios convergentes SOA/IMS, tanto para redes existentes como futuras, en este trabajo se propone una Lógica de Mediación, denominada MIDDIS, para tratar de hacer simple la complejidad de las Telecomunicaciones para el mundo de las IT, y con ello agilizar el desarrollo de cualquier tipo de servicio en el nuevo entorno de convergencia.

1.2. Escenario de Motivación

Para el desarrollo de servicios en un entorno donde IMS y SOA se traten como dos mundos separados, es posible construir una plataforma de aplicaciones IMS que de soporte únicamente a peticiones SIP, que se enlacen a un portal de servicios, y luego utilizar herramientas SOA para desarrollar la notificación de los servicios según los criterios de la Web 2.0 (Nolle, 2008). Esto permite la utilización de la admisión y el control de los recursos de IMS, y así se proveen funciones de gestión de entrada a usuarios y recursos, con el fin de prevenir la congestión y las interrupciones de los servicios. Sin embargo, la principal y mayor desventaja de este enfoque radica en que no se puede seguir controlando los servicios cuando salen del dominio SIP; en este caso los operadores de telecomunicaciones se convierten en simples transportadores de bits, al no controlar la señalización de los mismos, lo cual causa un impacto considerablemente negativo a su modelo de negocio actual y futuro, ya que en ese contexto no harían parte fundamental en la cadena de valor del nuevo mercado de servicios convergentes.

En la Lógica de Mediación, o Middleware, para la interacción de servicios basados en IMS y SOA (MIDDIS), a diferencia del anterior enfoque, se propone el desarrollo de servicios en un entorno donde IMS-SIP se pueden integrar con SOA-WS, con lo cual se busca que IMS llegue a representar realmente la estandarización de una SOA diseñada para aplicaciones de tiempo real, de gran escala, y segura (McHugh, 2006); propendiendo siempre por la participación de los operadores de telecomunicaciones en el control de la lógica de los servicios y por tanto buscando mantener su relevancia en la cadena de valor de los servicios convergentes.

1.3. Objetivos

Objetivo General

Proponer a los proveedores de servicios de telecomunicaciones una lógica de mediación basada en SIP/SOA para la interacción de servicios SOA/IMS, que permita el desarrollo de servicios convergentes.

Objetivos Específicos

- Crear una base de conocimiento alrededor de la creación y despliegue de servicios que utilizan las capacidades de las redes existentes y de IMS.
- Definir y desarrollar una lógica de mediación que permita la interacción de servicios basados en SOA e IMS.
- Desarrollar un piloto de Servicio Convergente que consuma capacidades de un Servicio SOA sobre una red IP y capacidades de un servicio de telecomunicaciones sobre una red IMS, con el fin de realizar la evaluación inicial del mediador creado.

1.4. Contribuciones de la Tesis y Estructura del Documento

Las contribuciones más importantes del presente trabajo de grado de maestría están estructuradas de la siguiente manera a lo largo de este documento:

En el Capítulo 2 se tratan los conceptos generales acerca de los servicios basados en IMS y SOA, las tecnologías más representativas en el área de los middleware java para telecomunicaciones, y se describen los trabajos relacionados más importantes en las áreas de la Convergencia de servicios IMS basados en SIP y Servicios Web (SOAP), y del Manejo de Sesiones en los Servicios Web. Los resultados obtenidos en este capítulo son: a) Un análisis comparativo de los antecedentes en el área de los Mediadores para Telecomunicaciones, orientado a seleccionar la herramienta más adecuada para dar soporte a la implementación de la Lógica de Mediación, y, por lo tanto, la más eficiente en el desarrollo de servicios, no solo de telecomunicaciones sino también de servicios basados en SOA; b) Un análisis y valoración de los antecedentes en la convergencia de servicios IMS, basados en SIP, y Web (SOAP), para el diseño de una la Lógica de Mediación basado en las mejores experiencias y resultados obtenidos hasta el momento en el área de la convergencia de servicios de telecomunicaciones (IMS) y de Internet (SOA); Un análisis y valoración de los antecedentes para el manejo de sesiones en los Servicios Web, que proporcione desde el mundo de las IT un soporte adicional a la gestión de las sesiones que se realiza, y es bien conocida, en el mundo de las telecomunicaciones.

En el Capítulo 3 se plantea la solución al problema de la interacción de servicios basados en IMS y SOA, a través de la caracterización de la misma y su arquitectura de referencia. Se definen los subsistemas que componen la arquitectura y la forma como se relacionan para constituir una Lógica de Mediación para la creación de servicios convergentes IMS/SOA. El resultado obtenido en este capítulo es: a) Una arquitectura de referencia que proporciona las funcionalidades para la mediación en la interacción de servicios basados en IMS y SOA.

En el Capítulo 4 se presenta una implementación de referencia, que corresponde a una instanciación de la arquitectura de referencia, y cuyo mecanismo de intermediación fue evaluado por medio de la creación de un piloto de servicio convergente con acceso a capacidades tanto del entorno de telecomunicaciones como del IT. Dicho piloto de servicio permitió analizar y probar el comportamiento de la solución propuesta en cada uno de los pasos requeridos dentro de un proceso de interacción de servicios basados en IMS y SOA. Los resultados obtenidos en este capítulo son: a)La arquitectura de la implementación de referencia, incluyendo las herramientas y tecnologías empleadas para instanciar la arquitectura de referencia; b) Un piloto de servicio convergente con acceso a las capacidades de un Servicio SOA sobre una red IP, y a las capacidades de un servicio de telecomunicaciones sobre una red IMS.

En el Capítulo 5, se plantean los aportes, las conclusiones, y los trabajos futuros.

Los principales resultados se han publicado o están en proceso de evaluación para su publicación en revistas indexadas de la siguiente manera:

- MIDDIS: Arquitectura de Referencia para la Interacción de servicios basados en SOA e IMS: artículo presentado a la Revista Ciencia e Ingeniería Neogranadina, de la Universidad Militar Nueva Granada, ISSN 0124-8170. El contenido del artículo se puede encontrar en el Anexo C.
- Mecanismos basados en SIP y SOA para el acceso a las capacidades de servicios convergentes IMS/SOA: artículo que va a presentarse a una revista indexada. El contenido del artículo se puede encontrar en el Anexo D.

Capítulo 2

SERVICIOS BASADOS EN IMS Y SOA

2.1. Conceptos Generales

2.1.1. IMS

La evolución del Sistema de Telecomunicaciones Móviles Universales (Universal Mobile Telecommunications System, UMTS), incorporó al mundo de las telecomunicaciones un nuevo subsistema para el desarrollo de servicios multimedia y usuario a usuario. Esta nueva arquitectura del 3GPP se denomina IMS (Camarillo y García, 2006) y se basa en SIP para la señalización. IMS permite la implementación de servicios de próxima generación a través de diferentes dispositivos y tecnologías de acceso, con la calidad y confiabilidad que los usuarios exigen en la actualidad (PC-NEWS, 2006).

IMS está siendo definido y diseñado como un sistema totalmente independiente del acceso, lo que permite un escenario de desarrollo de servicios sobre IP y servicios multimedia para usuarios conectados a través de la Línea de Abonado Digital Asimétrica (Asymmetric Digital Subscriber Line, ADSL), Redes Inalámbricas de Área Local (Wireless Local Area Network,WLAN), la Interoperabilidad Mundial para Acceso por Microondas (Worldwide Interoperability for Microwave Access, WIMAX), el Servicio General de Paquetes de Radio (General Packet Radio Service, GPRS), UMTS, o cualquier otro acceso, lo cual posibilita la convergencia, con el consiguiente beneficio para operadores (nuevos servicios y sinergias en las inversiones) y usuarios (facilidad de uso y acceso al mismo servicio desde cualquier terminal, en cualquier momento) (Carrascosa, 2005).

El objetivo de IMS es permitir la creación de nuevas aplicaciones multimedia capaces de suministrar al usuario final una experiencia de comunicaciones integrada, independientemente del tipo de aplicación, método de acceso a la red, e infraestructura de la misma, de modo que los usuarios finales tengan acceso a cualquier tipo de servicios (tanto tradicionales, como nuevos) desde cualquier sitio, en cualquier momento, y sobre múltiples tecnologías de acceso (Expocomm Argentina, 2005).

2.1.1.1. Arquitectura

La arquitectura de servicios IMS es una colección de funciones lógicas dividida en 3 capas: la de Acceso y Transporte, la de Control de Sesión, y la de Aplicaciones.

- **Acceso y Transporte**: conecta las diferentes redes de acceso con el corazón de la arquitectura IMS, mediante el empleo de pasarelas y servidores de control. Inicia y termina la señalización SIP, a través de la cual se puede configurar sesiones y realizar la provisión de servicios. (Mariotto, 2006) (Telefónica I+D, 2005)

- **Control de Sesión**: abarca los servidores de control de red para la gestión de las sesiones (establecimiento, modificación y liberación); contiene la Función de Control de la Sesión de Llamada (Call Session Control Function, CSCF), que constituye la entidad funcional clave, encargada de proveer el registro de los extremos finales y el enrutamiento de los mensajes de señalización SIP hacia los Servidores de Aplicaciones (Application Servers, AS) apropiados. (Mariotto, 2006) (Telefónica I+D, 2005) (Lucent Technologies, 2005)
  Esta capa contiene el Servidor Nominal del Suscriptor (Home Subscriber Server, HSS), que mantiene un perfil único de servicio por cada usuario final. Dicho perfil almacena toda la información de servicio y preferencias de un usuario en una localización central, razón por la cual las aplicaciones pueden compartir la información; adicionalmente este orden centralizado simplifica enormemente la administración de los datos de usuario (Mariotto, 2006). La comunicación entre la CSCF y el HSS se realiza utilizando el protocolo Diameter (Liu, y otros, 2006). Esta capa también incluye, tanto a la Función de Recursos Multimedia (Media Resource Function, MRF), encargada de los recursos multimedia, como a la Función de Control de la Pasarela de Medios (Media Gateway Control Function, MGCF) a cargo de la interoperabilidad SIP - H.248.
- **Aplicaciones**: contiene los Servidores de Contenido y AS encargados de proveer la lógica de servicio del usuario final. Tanto IMS como la señalización SIP son lo suficientemente flexibles como para soportar una variedad de aplicaciones, entre telefónicas y no telefónicas (Teléfonica I+D, 2005) (Lucent Technologies, 2005).

2.1.1.2. SIP

SIP ha existido desde 1995, y fue aprobado por la Fuerza de Trabajo de Ingeniería para Internet (Internet Engineering Task Force, IETF) en el año de 1999 bajo el RFC 2543 (Handley, y otros, 1999). La versión actual del protocolo está definida en el RFC 3261 (Rosenberg, y otros, 2002).

SIP es un protocolo de señalización del nivel de aplicación diseñado para el establecimiento, modificación, mantenimiento y terminación de sesiones interactivas sobre redes IP para diferentes tipos de aplicaciones. Además, aporta las funciones para el registro, enrutamiento de sesiones, identificación de usuarios y nodos, localización y disponibilidad de usuarios, negociación de las capacidades de comunicación y también habilita todo tipo de servicios suplementarios (Telefónica I+D, 2005) (Moreno, y otros, 2005). En el 3GPP se determinó la utilización de SIP como protocolo para conectar la gran mayoría de los nodos IMS entre sí, y con el resto de los elementos que componen una red de próxima generación. IMS utiliza también a SIP para establecer sesiones de multimedia y voz en Internet. Se lo eligió por su simplicidad, extensibilidad y su amplia disponibilidad (Pechuán, 2004).

Por definición SIP, no es un protocolo diseñado para una red o una aplicación específica. Para utilizar SIP, se debe definir un perfil de uso. Este último se utiliza como una plantilla, y proporciona un ambiente variado y flexible según los requisitos particulares. El perfil de SIP creado para IMS (SIP-3GPP) es uno de los más importantes en el ámbito de las telecomunicaciones, pues no solo involucra a las redes móviles sino también a toda la industria, y según los expertos actualmente es el más apropiado para las NGN (RADVISION, 2006). Para este perfil se crearon las cabeceras-P (P-headers), las cuales permiten realizar

tareas de tarificación, conocer la información de las redes en las cuales se cursa una llamada, la creación y gestión de nuevos servicios no multimedia, etc.

2.1.2. SOA y los Servicios Web

La tendencia actual del entorno de las IT es la integración de los sistemas de información existentes de una manera estándar, para hacer posible la interoperabilidad de las diferentes implementaciones. En este ámbito, los WS han surgido por su capacidad de proveer una interfaz de comunicación estándar entre sistemas heterogéneos.

Los WS son sistemas software que han sido diseñados para soportar una interacción máquina a máquina a través de la red, con características de interoperabilidad. Dicha interoperabilidad la obtienen a través de un conjunto de estándares abiertos basados en el Lenguaje de Marcado Extensible (eXtensible Markup Language, XML), tales como el Lenguaje de Descripción de Servicios Web (Web Services Description Language, WSDL), el Protocolo Simple de Acceso a Objetos (Simple Object Access Protocol, SOAP), y la tecnología de Descripción, Descubrimiento e Integración Universal (Universal Description, Discovery and Integration, UDDI), los cuales le proporcionan a un WS un enfoque común para su definición, publicación y uso.

Los WS combinan los mejores aspectos del desarrollo basado en componentes con los mejores aspectos de la Web. Al igual que los componentes, los WS representan funcionalidad de caja negra que se puede reutilizar, sin conocer los detalles de la implementación. Sin embargo, a diferencia de las tecnologías de algunos mediadores predecesores o actuales, las cuales implementan un modelo de componentes específico, los WS no se acceden a través de protocolos de un modelo de objetos determinado, tal como el Modelo de Objetos de Componentes Distribuidos (Distributed Component Object Model, DCOM), la Invocación Remota de Métodos (Remote Method Invocation, RMI), o el Protocolo Inter-ORB de Internet (Internet Inter-ORB Protocol, IIOP). En lugar de ello, los WS son accedidos a través de protocolos Web y formatos de datos universalmente aceptados, tales como el Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol, HTTP) y XML. (Curbera, y otros, 2001)

SOA es una estrategia de sistemas de información que organiza la funcionalidad, contenida en las aplicaciones, en servicios interoperables basados en estándares, que pueden ser combinados y reutilizados de forma rápida, y mediante interacciones con bajo acoplamiento, para cubrir las necesidades del negocio (Fernández, 2007) (Georgescu, 2007). Además, desacopla los recursos de la red subyacente del desarrollo y la prestación del servicio, lo cual simplifica la creación, composición y prestación de servicios misma, y por tanto permite a la comunidad de desarrolladores y proveedores de servicios una rápida innovación, puesto que el desarrollador no requiere entender la tecnología de la red subyacente con la cual se construyó el servicio. Utilizar un enfoque SOA para el desarrollo de aplicaciones compuestas e interoperables hace posible la combinación de servicios de múltiples organizaciones y dominios de aplicación, utilizando interfaces basadas en estándares, y hace que sea posible utilizarlos como parte de múltiples aplicaciones. (Telefónica, 2004)

SOA puede ser implementada utilizando cualquier tipo de infraestructura cliente-servidor y tecnología basada en servicios, pero lo más común es que se implemente con WS ya que éstos están basados en estándares, lo cual introduce una arquitectura orientada a servicios débilmente acoplada, extensible, muy adecuada a la infraestructura de Internet (Chou, y otros, 2006), y constituye la clave para la interconexión, integración, e interoperabilidad tanto de redes como de servicios.

2.1.3. IMS y SOA

Hoy en día, la mayoría de los servicios de telecomunicaciones están estructurados verticalmente. Además de la lógica del servicio y del negocio, cada servicio tiene sus funciones propias de gestión y ejecución, y sus respectivas interfaces hacia entornos como el Sistema de Soporte a la Operación (Operation Support System, OSS) y el Sistema de Soporte a la Facturación (Billing Support System, BSS), conllevando esto a que en una red no-IMS, los servicios son especificados y soportados por un nodo o un conjunto de nodos lógicos que realizan tareas especializadas para cada servicio específico. Como cada servicio es una isla, la única manera posible para lograr la interacción de servicios es a través de protocolos específicos para cada combinación. En ausencia de un marco de trabajo común para servicios, cada servicio se diseña, se prueba y se implementa desde el principio y debe ser mantenido y actualizado individualmente. (Andrinal, y otros, 2007)

Por el contrario, la filosofía de IMS propone la reutilización de muchas funcionalidades, lo que apunta a atacar el reto de creación y despliegue de servicios de manera mucho más rápida. En este sentido, y en una solución acorde con IMS, se trata de que los sistemas se diseñen para soportar múltiples aplicaciones. Esto significa que la misma infraestructura se pueda utilizar para nuevos servicios con el esfuerzo de crecimiento enfocado en el propio servicio y no en las características básicas. En consecuencia, sería muy simple y eficiente disponer de un nuevo servicio para un usuario IMS ya que la infraestructura básica necesaria (autenticación, autorización, tarificación, etc.) debería estar disponible.

Hoy en día, está claro que la evolución hacia IMS será una transición gradual, exigiendo la interacción de diferentes aplicaciones, tecnologías y protocolos de red. La clave para el éxito, en este entorno complejo y competitivo, es crear un medio común de prestación de servicios, capaz de permitir la interoperabilidad de todas las aplicaciones y de mediar las distintas tecnologías, sin preocuparse por el tipo de acceso o transporte. La transición de la capa de servicios de IMS hacia un entorno SOA, para alcanzar el despliegue de servicios de comunicaciones integrados, comenzó inicialmente en el dominio OSS/BSS, el cual está evolucionando hacia una arquitectura en la que los bloques de servicio de bajo acoplamiento y reutilizables (o habilitadores) se ensamblan en aplicaciones (Andrinal, y otros, 2007).

En las próximas secciones se describen algunas de las alternativas más sobresalientes que se han desarrollado hasta el momento para dar solución a la necesidad de la convergencia entre los WS y los Servicios de Telecomunicaciones basados en IMS (Baravaglio, y otros, 2005) (Mansutti, y otros, 2007).

2.1.4. Mediadores

2.1.4.1. Lógica de mediación

Los mediadores o Lógicas de Mediación son software, o sistemas, que permiten interacciones entre programas a nivel de aplicación, en un ambiente distribuido y sobre plataformas heterogéneas; además, abstraen de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando un modelo de programación de alto nivel (Carnegie Mellon, 2007). Como ejemplo de mediadores se tienen los AS, el Modelo de Objetos de Componentes (Component Object Model, COM) y DCOM de Microsoft, los WS, y CORBA.

2.1.4.2. Mediadores para Telecomunicaciones

2.1.4.2.1. Servlets SIP

Contenedores de Servlets SIP

Los Servlets SIP definen un modelo basado en contenedores, el cual es una extensión del bien conocido modelo de Servlets HTTP. Los Servlets SIP se diseñaron para simplificar el desarrollo de aplicaciones basadas en SIP y por lo tanto para incrementar la adopción de este protocolo. La primera versión pública de la especificación de la API de los Servlets SIP se lanzó a finales del 2002.

Los contenedores de Servlets SIP están desarrollados para trabajar solamente señalización SIP, y son buenos para aplicaciones sencillas. Todo lo que tiene que ver con SIP está bien especificado, y la pila maneja los hilos, el ciclo de vida y la replicación, pero todo aquello que no está relacionado con SIP no está especificado y es bastante incierto (por ejemplo, no controlan ninguno de los otros protocolos de red que se puedan llegar a utilizar) (Kmatveev, 2008). Sin embargo, los contenedores de Servlets SIP podrían cubrir las cuestiones de sincronismo (Mikhanov, 2008).

Estos contenedores exigen que los Servlets sean sin estado, lo que por un lado es bueno porque permite a los desarrolladores de contenedores elegir cualquier modelo de hilos de su preferencia, y por otro lado es una desventaja ya que los procesos de recuperación del estado podrían volverse bastante complejos para el desarrollador.

Servidores de Aplicaciones SIP

Actualmente dan soporte no solo a los Servlets SIP sino también a los Servlets HTTP y a los WS. Sin embargo, no se da soporte a la convergencia de servicios basada en la interoperabilidad HTTP/SIP/WS. No soportan directamente el registro de Servicios IMS/SIP y por ende tampoco la orquestación de Servicios IMS/SOA (SIP/WS). Es decir que en un AS SIP/IMS no existe una entidad funcional que por defecto registre las interfaces de un servicio IMS, de forma tal que otro servicio pueda utilizarlo en el proceso de creación de su lógica de negocio.

2.1.4.2.2. Entornos de Ejecución de Lógicas de Servicios Java

SLEE (OpenCloud, 2009a)

Un SLEE es un concepto bien conocido en la industria de las Telecomunicaciones. Significa un entorno de aplicaciones de alto desempeño, baja latencia y de procesamiento de eventos. Este

entorno se puede implementar (como así se ha hecho históricamente) con un conjunto de librerías software que proporcionan las facilidades que requiere el servicio en tiempo de ejecución.

Hoy en día, dado el progreso de las técnicas en ingeniería del software, estas capacidades pueden ser provistas por un AS. Sin embargo, hasta ahora los SLEE han sido propietarios, y por tanto incompatibles con otros SLEE, ya que cada uno de ellos ha proporcionado sus propios grupos de capacidades, con API propietarias. Esto ha significado que los servicios y las características no han sido portables.

JAIN (OpenCloud, 2009a)

Es una iniciativa de estándares de Java Sun, y hace parte del Java Community Process. Esta iniciativa tiene definido un conjunto de API de Java que hacen posible el desarrollo rápido de productos y servicios de comunicaciones de próxima generación basados en Java, incluyendo librerías para:

- Desarrollo de aplicaciones de alto nivel, tales como: API de proveedores de servicios y el Entorno de Ejecución de Lógica de Servicio (Service Logic Execution Environment, SLEE),
- control de la llamada,
- señalización a nivel de protocolos, tales como: SIP, el Protocolo de Control de Pasarela de Medios (Media Gateway Control Protocol, MGCP), y el Sistema de Señalización No.7 (Signaling System 7, SS7),
- etc.

JAIN SLEE (Lim, y otros, 2003)

JSLEE es el estándar de Java para el SLEE el cual surge de la tendencia actual hacia las plataformas abiertas, estandarizadas y basadas en componentes, y ofrece un entorno de ejecución de servicios en tiempo real que cumple con los requisitos del mundo de las Telecomunicaciones (Ivanov, 2006) (Cruz, 2005) (Falcarin y Venezia, 2008). La especificación incluye un modelo de componentes para la estructuración de la lógica de las aplicaciones de comunicaciones como una colección de componentes orientados a objetos y reutilizables, y para la composición de estos componentes en servicios de mayor nivel y más sofisticados (JAINSLEE.org, 2005a).

JAIN SLEE está diseñado como un modelo de componentes, especializado en aplicaciones orientadas a eventos y de forma asíncrona, para entornos de baja latencia y alto rendimiento, en donde las transacciones sean livianas y rápidas de concluir, es decir las características básicas de los servicios de telecomunicaciones. Entre otras de sus características se tiene que: reduce el tiempo de salida al mercado y el costo de desarrollo mediante la utilización de estándares, hace posible entornos de múltiples vendedores, proporciona un marco para la prestación de servicios portables, y hace una abstracción de la infraestructura subyacente a través de la utilización de Adaptadores de Recursos (Resource Adaptors, RA). (Maretzke, 2005)

En cuanto a los servicios que se pueden construir y desplegar sobre JAIN SLEE, se pueden citar como ejemplos los servicios tradicionales de inteligencia de red como VPN, traducción de números o menor costo de enrutamiento, servicios de localización, servicios de mensajería,

como cobro en tiempo real del servicio provisto por el Sistema de Mensajería Multimedia (Multimedia Messaging System, MMS) y servicios basados en señalización SIP, así como cualquier combinación de los anteriores. (Cruz, 2005)

JAIN SLEE, como paradigma de los entornos de nueva generación, rompe con la tradicional distinción entre los servicios de la Red Inteligente y los servicios IP. El concepto de RA permite la integración de diferentes tecnologías para obtener servicios innovadores y de nueva generación, así como el despliegue de servicios tradicionales basados en señalización, como la gestión de enrutamiento de llamadas. No existe un único campo de servicios adecuado para JAIN SLEE; por el contrario, una de las principales ventajas es su gran alcance, el cual es además ampliable mediante la incorporación de adaptadores de nuevos protocolos. Otra de las ventajas es su definición abierta, que elimina las barreras tradicionales de integraciones propietarias. (Cruz, 2005)

2.1.4.2.3. Resumen comparativo de los antecedentes en el área de los Mediadores para Telecomunicaciones

Para entender mejor los beneficios de cada uno de los modelos, y cómo podrían utilizarse de mejor manera, se puede considerar varias de las características conceptuales de las aplicaciones con el fin de compararlos y contrastarlos (JAINSLEE.org, 2005b) (Maretzke, 2008) (Gonzalez, y otros, 2009):

- **Arquitectura de la aplicación**: es la forma en que se estructura una aplicación en la lógica de negocio.
- **Estado de la aplicación**: es la forma en que se puede representar, acceder y proteger el estado de una aplicación.
- **Control de concurrencia**: es la forma en que pueden controlarse múltiples hilos de ejecución.
- **Facilidades disponibles para las aplicaciones**: son utilidades que el contenedor hace disponible para las aplicaciones.
- **Mecanismos de disponibilidad**: cómo se puede hacer uso de las características y conceptos del modelo para dar soporte a una disponibilidad continua.
- **Gestión**: es el mecanismo que proporciona el contenedor para gestionar las aplicaciones y al contenedor mismo.

En la Tabla 1 se realiza una comparación entre los SIP Servlets y JAIN SLEE, y se contrastan considerando las características nombradas anteriormente.

Tabla 1. Resumen comparativo de los antecedentes en el área de los Mediadores Java para Telecomunicaciones

SIP Servlets	JAIN SLEE
Arquitectura de la Aplicación	
- Basada en los Servlets HTTP. - La Unidad de la lógica de la aplicación es el Servlet. - No tiene un modelo estándar para la composición y reutilización.	- Arquitectura basada en componentes y orientada a objetos. - La unidad de la lógica de la aplicación es el Bloque de Construcción de Servicios (Service Building Block, SBB). - Da soporte para la composición y la

	reutilización.
Estado de la Aplicación	
- Los Servlets son sin estado. - Los estados compartidos pueden guardarse en objetos de sesión separados, según parejas de nombre y valor (Cadena, Objeto). - El estado compartido es visible a todos los Servlets con acceso a la sesión.	- El SBB puede ser con estado o sin estado. - El estado del SBB es privado, de tipo seguro, y es propiedad del mismo SBB (por ejemplo, un SBB es un objeto). - El estado compartido puede almacenarse en un *Contexto de Actividad* separado, a través de una interfaz de tipo seguro. - El acceso al estado compartido puede especificarse en el momento del despliegue.
Control de la Concurrencia	
- Solamente se gestiona la aplicación, por ejemplo se utilizan monitores Java.	- Se gestiona el sistema, y así, por ejemplo, hay aislamiento de las transacciones coexistentes.
Soporte a Protocolos	
- SIP y HTTP.	- Agnóstico a los protocolos. - Puede extenderse para dar soporte a protocolos adicionales y recursos externos. - Modelo de eventos consistente, sin importar el protocolo/recurso.
Facilidades (Utilidades para las aplicaciones)	
- Temporizador.	- Temporizador. - Trazado. - Alarma. - Estadísticas y Utilización. - Perfiles.
Mecanismos de Disponibilidad	
- El estado, que puede ser replicado, es gestionado por el contenedor (objeto de sesión). - No se maneja el contexto de la transacción para el procesamiento de los mensajes SIP. - Operaciones de estado no negociadas. - Las facilidades no son negociadas. - No está definido el modelo de fallas.	- El estado, que puede ser replicado, es gestionado por el contenedor (SBB, Persistencia Gestionada por el Contenedor (Container Managed Persistence, CMP), Contexto de la Actividad, facilidad de estado, etc.). - Se maneja el contexto de la transacción para la entrega de eventos. - Las operaciones de estado, gestionadas por el contenedor, son negociadas. - Las facilidades, por ejemplo los temporizadores, son negociadas. - Modelo de fallas bien definido y entendido por medio de transacciones.
Gestión	
- No existe la definición de mecanismos estándares de gestión.	- Se definen interfaces de gestión estándares, basadas en JMX (Java Management Extensions). - Independiente del protocolo de Gestión. - Interfaz para la gestión de aplicaciones, incluyendo: ciclo de vida, actualizaciones, perfiles, trazado, etc. - Interfaz para la gestión del ciclo de vida del SLEE.

2.2. Trabajos Relacionados

2.2.1. Antecedentes en la Convergencia de servicios IMS basados en SIP y Servicios Web (SOAP)

La arquitectura IMS define que las invocaciones entrantes a los servicios se lleven a cabo a través de sesiones SIP, las cuales son continuas y de corta vida, comparadas con las sesiones y las transacciones de larga vida utilizadas por un gran número de tecnologías mediadoras de alto nivel como los WS (Levenshteyn y Fikouras, 2006). De esta manera, para proveer a clientes de WS externos acceso a los servicios del dominio IMS se requiere gestionar la invocación del WS al interior de las sesiones SIP. Sin embargo, hoy en día muchos de los servicios existentes en el dominio del operador aún no se basan en SIP, y tales servicios utilizan protocolos y tecnologías alternativas, en muchos casos propietarias.

SIP (perfil para IMS), y SOAP (de los WS), son protocolos surgidos del mundo de Internet, ideales para entornos descentralizados y distribuidos, donde muchos componentes colaboran en la provisión de características y funcionalidad. En este contexto: i) SOAP es excelente como protocolo liviano para el intercambio adaptable de información, en un entorno descentralizado y distribuido, además es un protocolo modular y extensible; entre sus limitaciones se tiene que no está orientado a la conexión (Tikkanen, 2006), lo que significa que no puede mantener el estado de la sesión y la información de la transacción entre WS (Dong y Newmarch, 2005). ii) SIP por su parte, posee las capacidades requeridas para el descubrimiento de componentes, duración y control de sesiones (Deason, 2001), y una de sus principales características es la separación del flujo de los mensajes de señalización del flujo de los datos del usuario; sin embargo, a pesar de su simplicidad, flexibilidad y extensibilidad para el manejo de sesiones, no está basado ni en XML ni en WS (Chou, y otros, 2006).

A pesar de las diferencias y limitaciones nombradas, si se toman como ventajas las bases tecnológicas y la filosofía de diseño que SIP y SOAP comparten, en lugar de tratarse de protocolos incompatibles, su combinación puede resultar eficiente para la construcción de servicios sofisticados y convergentes, tanto para el entorno IT como para el de las Telecomunicaciones. Por lo tanto, lo que se busca en un escenario real de implementación es que SOAP haga posible WS avanzados y SIP permita que estos se utilicen al interior de aplicaciones integradas de comunicaciones (Deason, 2001). En este contexto, y bajo la tesis de la posibilidad de compatibilidad entre SIP y SOAP, desde el punto de vista de los protocolos pueden surgir diversas alternativas para su combinación. En las próximas secciones se describen algunas de ellas, las cuales se han desarrollado para dar solución a la necesidad de la convergencia entre los WS y los Servicios de Telecomunicaciones, constituyéndose así en base fundamental para la propuesta de solución a la problemática planteada en el presente trabajo.

2.2.1.1. TekSCIM

El Gestor de la Capacidad de Interacción de los Servicios (Service Capability Interaction Manager, SCIM), introducido en la arquitectura de IMS del 3GPP (TS 23.002), se define como el elemento de la capa de aplicación de IMS que hace posible la integración de las capacidades de distintos servicios para poder obtener servicios enriquecidos. TekSCIM es la solución SCIM de Tekelec (Tekelec, 2007).La plataforma TekSCIM está hecha específicamente para la

orquestación y mediación de servicios entre redes que utilizan múltiples tecnologías. En (Tekelec, 2007) se examina cómo los operadores pueden utilizar el servicio de mediación y orquestación para crear una arquitectura flexible para la prestación de servicios enriquecidos y compuestos, en redes no-IMS, IMS e híbridas. Sin embargo, a pesar de la importancia del SCIM, las especificaciones del 3GPP respecto a IMS contienen muy poca información sobre dónde y cómo se debería implementar su funcionalidad. Adicionalmente, no hay estándares que definan la integración de servicios y aplicaciones, o cómo orquestar e intermediar múltiples servicios para la creación de paquetes de servicios. En gran parte, dicha falta de detalles se debe a la controversia de la industria acerca del lugar en la red en el cual debería estar el SCIM.

2.2.1.2. SPICE

Este proyecto aborda la creación de servicios convergentes para IMS, utilizando la Plataforma de Servicios para un Entorno de Comunicaciones Innovador (Service Platform for Innovative Communication Environment, SPICE) (Tarkoma, y otros, 2008). Esta plataforma diseña, desarrolla, y hace las pruebas de un entorno de creación y ejecución de servicios móviles innovadores para redes que van más allá de la 3G. Aborda los retos en la provisión de servicios convergentes, mediante el desarrollo de un marco de trabajo para el rápido desarrollo de nuevos servicios móviles, en un mundo donde la convergencia de los servicios de telecomunicaciones y los servicios basados en Internet ha adquirido un interés considerable. SPICE se basa en las IT, utilizadas en el desarrollo de servicios de Internet, incluye habilitadores del futuro sistema IMS, combina diversas tecnologías clave tales como: un sistema mediador basado en componentes semánticos, intermediación de servicios y mecanismos de mediación, y gestión del ciclo de vida, entre otros. Adicionalmente, especifica la plataforma para el terminal, la plataforma del proveedor de servicios y un entorno para la creación de servicios.

2.2.1.3. SOAP/SIP binding

SOAP es un mecanismo de transporte neutral para el intercambio de mensajes. Esto significa que cualquier protocolo de transporte o de la capa de aplicación puede transportar un mensaje SOAP, en cuanto cumpla el conjunto formal de reglas que se definen para ello. Cuando se utiliza a SOAP sobre SIP, es decir en el plano del usuario (datos), se hace posible la transmisión de los mensajes SOAP al interior del cuerpo de un mensaje SIP (Gehlen, y otros, 2006). Esta alternativa corresponde al caso predeterminado de transmisión de mensajes SOAP al interior de un mensaje HTTP, fue desarrollada por la empresa Ubiquity Software y propuesta a la IETF. En ella se introduce un nuevo método SIP: "`SERVICE`", con el cual es posible llevar mensajes SOAP como carga útil de un mensaje SIP (Deason, 2001). En algunos de los escenarios reales en los que se aplicó esta alternativa se demostró, por ejemplo, que SIP se puede utilizar para escoger el dispositivo adecuado del destinatario y SOAP le puede indicar al servicio los canales que se van a utilizar (Tikkanen, 2006). No obstante la importancia de este nuevo método SIP, así como de otros intentos de aprobación de métodos similares, como es el caso del método SIP del tipo "`DO`" (Tsang, y otros, 2000), propuesto por la empresa Telcordia Technologies, y que se incluye dentro del proyecto HomeSip (IMS Laboratory, y otros, 2006), para la utilización del protocolo SIP como protocolo de control de una casa domótica, nunca se han llegado a aceptar alguna de éstas propuestas.

2.2.1.4. Creación de Servicios Web P2P y móviles utilizando SIP

SIP se define como un protocolo de señalización en la capa de aplicación, por lo que es posible su utilización en el plano de control (señalización) en paralelo con SOAP, que a su vez se encuentra en el plano del usuario (datos), como se propone en la alternativa del proyecto presentado en (Gehlen, y otros, 2006). La separación estricta entre los planos de usuario y de señalización tiene ventajas con respecto al diseño de los protocolos, al diseño del software de comunicación, y al desempeño (por ejemplo, se reduce la carga de la red con la infraestructura SIP). En este sentido, en el Plano del Usuario se pueden transmitir los datos específicos del usuario de la aplicación a través de SOAP, por encima de los diversos protocolos de Internet subyacentes, y en el Plano de Control (Señalización), se puede utilizar SIP para transmitir los mensajes de señalización de la capa de aplicación.

2.2.1.5. WSIP: Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP

La combinación de SIP con los WS proporciona un nuevo paradigma de comunicación convergente. Otra alternativa existente en esta área es el Terminal WS/SIP para Comunicaciones Multimedia y Multimodales sobre IP (Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP, WSIP), presentado en (Liu, y otros, 2004) (Chou, y otros, 2006). En esta solución se presenta un enfoque para la convergencia de servicios de comunicaciones sobre IP, basado en el concepto de WSIP: la combinación de SIP con los WS. De acuerdo a este enfoque, de doble pila, cada nodo WSIP es tanto un nodo SIP, que se comunica en el mundo de SIP a través de ésta señalización, y también es un nodo de WS (SOAP), que proporciona un entorno nativo y genérico para la integración de servicios.

Para integrar a SIP en las transacciones empresariales a través de XML, se han hecho esfuerzos para embeber el control/respuesta del servicio basado en XML, como parte de los cuerpos de los mensajes originales de petición de SIP tales como INVITE, INFO, SUBSCRIBE, NOTIFY. Este enfoque pragmático encuentra varios problemas, debido a que el protocolo SIP carece de la semántica que dé soporte a tal uso.

Otro intento consiste en embeber el control/respuesta del servicio basado en XML en el Protocolo de Descripción de la Sesión (Session Description Protocol, SDP), como parte del INVITE. Sin embargo, SDP se limita a una sintáxis de descripción (con reglas de análisis sintáctico estrictas), por lo cual no proporciona un rango completo de capacidades de negociación para el control/respuesta, y no está diseñado para extenderse fácilmente. De nuevo, esto se sale del uso actual de SDP en el mensaje INVITE. Además, el Agente de Usuario (User Agent, UA) SIP receptor puede ignorar el SDP del INVITE y responder con su propio SDP para la configuración de la llamada.

La idea principal detrás del enfoque de WSIP es no complicar más el protocolo de señalización SIP, sino exponer en el mundo de los WS un UA SIP como un nodo SOAP. Al hacerlo, se separa la señalización del entorno de integración de servicios, a través del modelo de bajo acoplamiento de los WS. Esto proporciona un entorno extensible y estándar basado en XML para la integración de servicios, a través de la utilización de SOAP/WSDL, lo cual permite la fácil integración de SIP en las transacciones empresariales. En este enfoque, un terminal WSIP

puede integrarse remotamente con cualquier aplicación autorizada y de una manera distribuida, a través del método estándar de los WS. No hay necesidad de cambiar el protocolo de señalización SIP, y no se requiere que una aplicación posea un UA SIP con el fin de transmitir el contenido y el control del servicio. La ubiquidad del entorno de integración de servicios se logra a través del enlace de SOAP con HTTP y de la interfaz común de descripción de servicios, WSDL.

2.2.1.6. WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP

La comunicación convergente de multimedia y voz sobre IP es un área de investigación activa debido a su alta demanda y a la naturaleza quebrantadora de ésta tecnología al proporcionar un medio alternativo, con nuevos y mejores beneficios, tanto para usuarios como proveedores de red y de aplicaciones, para los servicios de comunicaciones sobre IP. Por ello, muchos enfoques se han propuesto, y continúan proponiéndose, para no obstaculizar el poder de las comunicaciones basadas en IP. Uno de esos enfoques es WSIP (Servicios Web + SIP) el cual es una solución de doble pila que utiliza los WS para separar la integración de servicios de la señalización, y de la transmisión multimedia. WSIP utiliza un protocolo de la capa de aplicación superpuesto, como por ejemplo HTTP, y SIP, para la comunicación y para las extensiones del servicio. Sin embargo, el enfoque de doble pila de WSIP puede que no sea posible en las plataformas de WS o ni siquiera deseable donde es viable. En WSIP, SIP hace parte de su pila doble, y en consecuencia tiene que implementar SIP y tratar con muchos de los asuntos heredados que se relacionan con SIP.

En consecuencia a lo anterior, en (Chou, y otros, 2006) se introduce y define un protocolo de WS denominado Protocolo de Inicio de Servicios Web (Web Service Initiation Protocol, WIP) para comunicaciones multimedia y de voz sobre IP. Este enfoque se encuentra basado completamente en WS, y en una única pila de WS para la señalización de los servicios de comunicación.

WIP consiste en un conjunto de operaciones de WS para el inicio y el establecimiento de servicios de comunicaciones convergentes (por ejemplo multimedia, mensajería instantánea, voz, etc.) sobre IP. WIP hereda de SIP el principio de separación de la transmisión de multimedia de la de señalización; sin embargo se basa solamente en una pila de WS para proporcionar un protocolo de señalización de comunicaciones completamente equipado, dejando por fuera de acción a la necesidad de convergencia con aquellas aplicaciones y servicios, basadas en la señalización SIP del dominio de las Telecomunicaciones.

WIP abre un nuevo paradigma de WS basados en comunicaciones VoIP, el cual puede extenderse y se puede integrar fácilmente en soluciones extremo a extremo basadas en SOA. El enfoque genérico de WS que se utiliza en WIP supera muchas de las limitaciones que de otra manera serían difíciles de alcanzar con los métodos de comunicación que no se basan en WS y que se utilizan hoy en día, pero es la misma fortaleza de este enfoque la que limita su campo de acción ya que desconoce la amplia adopción que se ha hecho de SIP para el soporte a la señalización en las NGN.

2.2.1.7. Invocación de Servicios Web por medio de la señalización SIP

En la alternativa presentada en (Fornies, y otros, 2007) se implementó la invocación de WS por medio de la señalización SIP, donde esta última es aplicada para resolver los problemas de disponibilidad e identificación de los WS.

Actualmente es posible utilizar un dispositivo móvil como un consumidor de WS, para lo cual es necesario crear un cliente en la aplicación del terminal y consumir el método correspondiente del WS para obtener los datos deseados. Sin embargo, cuando se consume una operación de un servicio, se deben tener en cuenta dos aspectos que pueden obstaculizar la interacción con dicho servicio. Por un lado, el dispositivo que aloja el servicio puede no estar disponible en el momento en que se hace la comunicación. El otro problema que se puede encontrar es que la identificación del servicio, que está contenida en su descriptor, corresponda a un espacio o dominio cuyo acceso no está disponible o es desconocido (como por ejemplo, un dominio privado de red).

Actualmente no hay una solución técnica para el problema de la disponibilidad de los servicios, en entornos móviles y distribuidos. El protocolo SIP y los registros hacen posible la determinación de la disponibilidad de los dispositivos, pero no se ha resuelto la disponibilidad de los servicios y de las operaciones que ellos proporcionan.

En (Fornies, y otros, 2007) se logra el acceso a los servicios proveídos por dispositivos móviles con acceso a redes IP, mediante la tecnología de los WS, utilizando el protocolo SIP como señalización. Básicamente, el dispositivo que provee el WS: está identificado por una dirección o URI, la cual es una dirección SIP, y se descubre y localiza mediante el inicio de sesiones SIP.

2.2.1.8. Akogrimo: hacia una Arquitectura IMS extendida

La tendencia de la fusión de las infraestructuras de Telecomunicaciones con las infraestructuras tradicionales de las IT está en curso. En el área de las Telecomunicaciones, IMS es una plataforma de servicios promisoria, pero su rango de aplicabilidad está ligado a servicios con capacidades SIP. Por tanto, se cree que integrando SOA e IMS se ampliará el rango de aplicabilidad de IMS, virtualizando todos los recursos de IMS a través de los WS, y para ello en (Jähnert, y otros, 2007) (Villagrá y Wesner, 2007) (Loos, y otros, 2005) se describe una solución para fusionar las aplicaciones basadas en SIP y SOAP.

Los autores de (Jähnert, y otros, 2007) creen firmemente que estos conceptos se necesitan con urgencia, para poder ser finalmente exitosos en el mercado. Por tanto, se requiere del concepto de la integración SIP-SOAP para poder llevar servicios de valor añadido basados en los WS a la red móvil.

La arquitectura orientada a servicios de Akogrimo suministra una plataforma abierta de servicios que da soporte a todo el ciclo de vida de una aplicación organizacional distribuida y transversal. Empezando desde la Identificación de los Proveedores de Servicios necesarios, se da soporte a la articulación de la colaboración y la implementación de una Organización Virtual Operativa (Operative Virtual Organization, OpVO), incluyendo diferentes proveedores por medio de Acuerdos a Nivel del Servicio (Service Level Agreements, SLA), y la promulgación de flujos de trabajo, enriquecidos semánticamente, en estas coaliciones creadas bajo demanda.

Durante esta etapa de operación, y a lo largo del tiempo, los participantes pueden estar sometidos a una evolución, o inclusive podrían ser reemplazados o removidos completamente del proceso de colaboración. Este reemplazo puede ser causado por azares del contexto, por ejemplo un participante cambia, de manera transparente, a un terminal con diferentes capacidades, o puede cambiar la disponibilidad del ancho de banda en una red de acceso móvil. Además, es un concepto único la integración en un único flujo de trabajo de los recursos virtualizados, tales como el cómputo o el almacenamiento, servicios de las aplicaciones, y también las comunicaciones multimedia, y la virtualización de los recursos de red. El último objetivo de Akogrimo es proporcionar una plataforma abierta de aprovisionamiento de servicios para los Operadores de Red y los Proveedores de Servicios.

La solución propuesta, que ha sido implementada en un prototipo y demostrada en el proyecto Akogrimo, es una base prometedora que ha sido fundamental para la propuesta de solución del presente trabajo.

2.2.1.9. FOKUS Open SOA Telco Playground, un Entorno de Telecomunicaciones basado en SOA

En septiembre de 2007 el FOKUS Open SOA Telco Playground se abrió oficialmente para encargarse de la Investigación y el Desarrollo en la capa de aplicación de IMS, de cara a los nuevos habilitadores IMS y su orquestación (con elementos como el SCIM, para la intermediación de servicios), de los principios de SOA y de las API de la Web 2.0.

El trabajo que se está llevando a cabo en (Carvalho de Gouveia y Magedanz, 2008) se resume a continuación:

- Integración de los servicios de telecomunicaciones y los de la Web 2.0 bajo la utilización de los principios de SOA, centrándose en ambos sistemas finales así como en la orquestación de servicios.
- Desarrollo de intermediadores avanzados de servicios para servicios IMS (basados en SIP) y no IMS (basados en HTTP).
- Desarrollo y extensión, basado en SOA, de servidores de aplicaciones IMS.
- Desarrollo de prototipos de servicios SOA en la parte superior de IMS.
- Diseño y Desarrollo de una solución de gestión extensible para entornos NGN/IMS basados en SOA, incluyendo los sistemas de aprovisionamiento de servicios para entornos IMS basados en SOA, sistemas de monitoreo y gestión de fallas.

Su misión es apoyar a las empresas de telecomunicaciones en el diseño de su estrategia individual SOA y en la implementación de soluciones específicas, así como en el desarrollo de nuevos procesos de negocio, al tiempo que mejoran los flujos de trabajo y hacen uso de los sistemas heredados dentro de una empresa que se extiende hacia SOA. (Magedanz, y otros, 2008)

2.2.1.10. WIMS 2.0

En paralelo con la evolución de los servicios de telecomunicaciones, el mundo de Internet ha presenciado la revolución de la Web 2.0, cuya fórmula de éxito se ha basado en situar al

usuario en el eje central del desarrollo de servicios. En la filosofía de la Web 2.0, el usuario es el protagonista, participa en la definición de servicios y usa Internet como una plataforma de la que obtiene los servicios que desea, mezclando funcionalidades (o haciendo mashups) y contenidos de diferentes fuentes. La iniciativa WIMS 2.0 (Al-Begain, y otros, 2009) (Moro, y otros, 2008), promovida por Telefónica y desarrollada técnicamente por Telefónica I+D, analiza estas tendencias y, en concordancia con los planteamientos de la iniciativa Telco 2.0 (Telco 2.0, 2009), plantea que: si los operadores de telecomunicaciones desean obtener nuevos servicios exitosos, deben abrir sus redes y ser proveedores de plataformas para servicios de Terceros, para así conseguir que lo demandado por los consumidores sea lo ofertado por el propio operador.

WIMS 2.0 considera que IMS, junto con las capacidades de telecomunicación de los operadores, conforma una plataforma apropiada para conseguir la convergencia entre el mundo de las telecomunicaciones y el mundo de Internet, situando al usuario en el centro y disponiendo los medios para crear nuevos servicios, ricos y variados, que el usuario define a su gusto e incluso colabora en su creación.

Para llegar a la convergencia deseada se definen dos vertientes complementarias: Ofrecer capacidades del operador a la Web 2.0, y Utilizar la Web 2.0 para enriquecer los servicios del operador. Con la primera estrategia, se pretende ofrecer funcionalidades IMS de la red de Telecomunicaciones hacia la Web 2.0, y, en consecuencia, el servicio final podría ser ofrecido por un Tercero situado en la Web 2.0, aportando el operador un valor añadido y manteniéndose con un rol activo dentro de la cadena de valor, más allá de ser un mero proveedor de conexión. La segunda estrategia complementa a la primera y plantea los medios técnicos para que el operador de Telecomunicaciones explote las posibilidades de la Web 2.0 en sus propios servicios. La principal aproximación seguida para llevar a la práctica los conceptos de WIMS 2.0 ha sido la exposición de capacidades del operador de Telecomunicaciones hacia la Web 2.0, mediante el API para la Transferencia de Estado Representacional (Representational State Transfer, REST) (Costello, 2002). No obstante, dependiendo de factores como la complejidad del caso de uso o el escenario de aplicación, desde la iniciativa WIMS 2.0 no se descarta el uso del API para SOAP, como por ejemplo las API para ParlayX.

2.2.1.11. Definición de la Arquitectura de una Plataforma de Prestación de Servicios, utilizando conceptos genéricos de IMS y SOA

Actualmente, la red de Telecomunicaciones está evolucionando hacia una Plataforma de Prestación de Servicios (Service Delivery Platform, SDP), que maneje la convergencia de las Telecomunicaciones e Internet/IT. Sin embargo, las SDP no tienen una arquitectura estandarizada. Es por ello, que en (Rolan y Hu, 2008) se define la arquitectura de una SDP basada en conceptos genéricos extraídos de IMS (Telecomunicaciones) y de SOA (Internet), ya que por sí solas, tanto IMS como SOA, proveen arquitecturas limitadas de plataformas de servicios que dependen de las tecnologías subyacentes.

IMS no define una arquitectura completa para una plataforma de servicios. Mejor que ello propone una capa de servicios limitada que está constituida, entre otros, por AS SIP, que alojan este tipo de aplicaciones. Estas últimas se comunican con las entidades funcionales de la

red utilizando mensajes SIP. La capa de servicios de IMS realiza la abstracción de las entidades funcionales y proporciona un punto de referencia basado en SIP, en la cual se despliegan diversas plataformas de servicios basadas en estándares. Entre ellas se incluyen la de la Red Inteligente (Intelligent Network, IN), Parlay (Parlay, 2009), Parlay X (Parlay X, 2009), SCIM y plataformas de servicios basadas en SIP. De esta forma, las plataformas de servicios utilizan la capa de servicios para acceder a las funciones de red subyacentes.

La arquitectura de la plataforma de servicios de IMS es la integración de la capa de servicios y de las plataformas de servicios. Por tanto, para utilizar dicha arquitectura en la definición de la arquitectura de la SDP se extrajeron sus conceptos genéricos.

Por otra parte, para emplear a SOA en la definición de la arquitectura de la SDP, se utilizó su definición de neutralidad tecnológica, que es la SOA Genérica (Generic SOA, GSOA). La GSOA se utilizó para definir arquitecturas de plataformas de servicios independientes de los detalles de las tecnologías, la distribución y la implementación. En este sentido, la SOA basada en los WS es una implementación de GSOA.

Como resultado de lo anterior, la arquitectura de la SDP integra la arquitectura de la plataforma de servicios de IMS y GSOA.

2.2.1.12. Modelo de Comunicaciones basado en SIP para Servicios Web en Tiempo Real

En (Cheng, y otros, 2008) se presenta un panorama general del reciente desarrollo en el área de las Telecomunicaciones, centrado en las tecnologías clave que se pueden aplicar a la habilitación de WS de comunicaciones (Rezabeigi, y otros, 2008) (Venezia y Falcarin, 2006). En particular, se presenta el Modelo de Comunicaciones basado en SIP para Servicios Web en Tiempo Real (Real-Time Web Services Communication Model, RT-WSCM).

En el dominio de las Telecomunicaciones, las transacciones con estado se basan generalmente en una sesión que maneja el estado a través de un contexto, y así, por ejemplo los modelos de llamadas con estado se utilizan para mantener el progreso de cada sesión. Por su parte, los métodos actuales de los WS están diseñados principalmente para la integración de servicios y se basan típicamente en un patrón de interacción de petición/respuesta sin estado. No obstante, muchos servicios estándares de telecomunicaciones, tales como llamada en espera, conferencias y pagar por llamar requieren que se intercambien muchos mensajes, y normalmente se realiza el control de la sesión por el tiempo que ésta dure. Debido a esto, en (Cheng, y otros, 2008) se proponen y diseñan máquinas de estados mejoradas para el manejo de las llamadas en el RT-WSCM, trabajando con motores asíncronos basados en eventos como JAIN SLEE, que es una tecnología diseñada no solo para el entorno de las Telecomunicaciones sino que también es capaz de integrar a los WS.

2.2.1.13. Resumen comparativo entre los antecedentes en la Convergencia de servicios basados en IMS (SIP) y Web (SOAP), y MIDDIS

En la Tabla 2 se realiza una comparación entre las alternativas presentadas para el soporte a la Convergencia de servicios IMS (SIP) y WS (SOAP), y MIDDIS.

Tabla 2. Resumen comparativo de los antecedentes en la Convergencia de servicios IMS (SIP) y WS (SOAP), y MIDDIS

Proyecto	Elementos considerados en MIDDIS	Elementos diferenciadores con MIDDIS
TekSCIM	Conceptos de los trabajos de Tekelec relacionados con la mediación de servicios convergentes en redes pre-IMS, IMS e híbridas	En MIDDIS se propone la definición de aproximaciones arquitecturales basadas en SOA e IMS para facilitar la interacción de servicios convergentes, en lugar de utilizar el concepto del SCIM y su correspondiente implementación de Tekelec, ya que se considera que el TekSCIM no debería estar ubicado en la capa de control sino en la de aplicación.
SPICE	Fundamento importante, tanto teórico como práctico, en el área que trata los mecanismos de la mediación de servicios.	Los objetivos de ambos trabajos difieren ya que por un lado, el objetivo del presente proyecto no es crear un marco de trabajo completo para el desarrollo de servicios convergentes como lo propone SPICE, sino que aborda la convergencia de servicios a nivel de interacción bajo un enfoque arquitectónico que incluye aspectos tanto de SOA como de IMS, para así tener en cuenta la tendencia en la creación de servicios basados en SOA, proveniente del mundo de las IT y las capacidades de convergencia de Telecomunicaciones proporcionadas por IMS.
SOAP/SIP binding	Transmisión de los mensajes SOAP al interior del cuerpo de un mensaje SIP.	En MIDDIS no se considera la utilización de los métodos SIP “SERVICE” o “DO”, ya que estos no están estandarizados.
Creación de Servicios Web P2P y móviles utilizando SIP	SIP para la transmisión de los mensajes de señalización y control, y SOAP para la transmisión de los datos específicos del usuario de la aplicación.	En MIDDIS se considera la utilización de la interacción SIP/SOAP para transmitir tanto los mensajes de señalización como los mensajes de control, y no solamente la utilización de SIP para estas tareas.
WSIP	Provisión de un entorno nativo y genérico para la integración de servicios, por medio de los WS.	La implementación de un enfoque de doble pila, como el terminal WSIP, para la integración de servicios SIP y Web, implica que los desarrolladores de WS necesiten entender muchos de los asuntos heredados que se relacionan con SIP, toda vez que necesite implementar un servicio convergente SIP/WS. En MIDDIS, el proceso de convergencia de servicios se centraliza en la lógica de mediación, la cual se desarrolla una sola vez y se despliega para su uso, reutilización y extensión. Los inconvenientes a los que hace referencia el enfoque WSIP, al tratar de embeber directamente el control/respuesta de los servicios dentro del protocolo SIP, quedan subsanados a través del mecanismo de integración de servicios basado en SIP y WS de MIDDIS, donde ésta última es la única que debe encargarse de procesar simultáneamente tanto los protocolos como las extensiones de SIP y los WS.

WIP	Mantenimiento del estado y la gestión de la sesión en los WS, en base a las nuevas especificaciones desarrolladas de los servicios Web tales como WS-Addressing, WS-Session, WS-ReliableMessaging, WS-Eventing, etc., haciendo que se avance de los servicios Web convencionales de petición y respuesta a la integración de servicios de dos caminos, es decir una interacción de servicios Web full duplex sobre IP.	Debido a que este enfoque se encuentra basado completamente en WS, y en una única pila de WS para la señalización de los servicios de comunicación, no existe la posibilidad de su utilización en la interoperación con Servicios de una red IMS, que requieren como mínimo de la gestión de la señalización del acceso a los servicios a través de SIP.
Invocación de Servicios Web por medio de la señalización SIP	Invocación de WS y resolución de los problemas de disponibilidad e identificación de los WS por medio de la señalización SIP.	Los WS no se identifican, descubren y localizan por medio de direcciones SIP, sin embargo la interacción entre servicios basados en IMS y SOA se puede realizar mediante la iniciación de sesiones SIP, a través de la Lógica de Mediación de MIDDIS.
Akogrimo	Integrando SOA e IMS se ampliará el rango de aplicabilidad de IMS: virtualizando todos los recursos de IMS a través de los WS, y por medio de la fusión de las aplicaciones basadas en SIP y SOAP. El enfoque adoptado por Akogrimo, que se va a utilizar en MIDDIS, consiste en utilizar a SIP como protocolo de señalización (localización de servidores, inicio de sesiones), y luego utilizar a SOAP en esas sesiones para las invocaciones normales a WS. De esta forma, también se logra que los proveedores de servicios y sus servicios se puedan localizar dinámicamente.	La solución de Akogrimo está orientada a las infraestructuras de Servicios basadas en Mallas (Grid), y utiliza algunas de las soluciones y conceptos surgidos de la comunidad Grid/Web Services. Aunque se utilizan algunos de los conceptos de su propuesta para fusionar a SIP y SOAP y por consiguiente para hacer posible la utilización de WS en IMS, los conceptos de Organización Virtual (Virtual Organisation, VO) o de Organización Virtual Dinámica Móvil (Mobile Dynamic Virtual Organisation, MDVO) (Jähnert, y otros, 2007) utilizados en la arquitectura conceptual de Akogrimo, no se van a utilizar. Por otra parte, a pesar de que la arquitectura de Akogrimo no está orientada a sustituir la infraestructura IMS sino a mejorarla, se observa como en el área de la gestión de los recursos de red limita su red de transporte a una NGN totalmente IP, y no tiene en cuenta a redes de transporte como UMTS, por ejemplo, lo que hace a su solución un obstáculo en la convergencia de redes pre-IMS e IMS.
FOKUS Open SOA Telco Playground	Aplicación de los principios de SOA y de las API de la Web 2.0 en IMS, para obtener un entorno de Telecomunicaciones basado en SOA. Se considera que solamente una SDP basada en SOA, en la parte superior de IMS, proporciona los servicios necesarios para justificar las inversiones en una NGN.	A pesar de que algunos fabricantes de la industria de las telecomunicaciones quieren mantener la complejidad en IMS, los proveedores de las IT quieren ayudar para que IMS sea más simple para los desarrolladores de servicios (McHugh, 2006), y es por ello que una de las funciones más importantes de MIDDIS es la abstracción de los recursos de la red subyacente, para que estos sean tratados y manejados por desarrolladores no necesariamente expertos en Telecomunicaciones. En este sentido, los

		elementos actualmente utilizados en la capa de servicios de IMS que tiene en cuenta el Open SOA Telco Playground son considerados como simples recursos de red, cuyos eventos serán atendidos por un elemento de más alto nivel como MIDDIS, utilizando el enfoque de JAIN sobre IMS (Magedanz, 2007), en los procesos de interacción de Servicios Convergentes IMS/SOA.
WIMS 2.0	Obtener la convergencia entre el mundo de las Telecomunicaciones y el mundo de Internet (principal objetivo buscado por la iniciativa WIMS 2.0), situando al usuario en el centro y disponiendo los medios para crear nuevos servicios, ricos y variados. En MIDDIS se propone que el entorno resultante se centre en el usuario, y no en los servicios como ocurre actualmente, lo cual va de acuerdo con el objetivo de WIMS 2.0.	La principal aproximación seguida para llevar a la práctica los conceptos de WIMS 2.0 es la exposición de las capacidades del operador de Telecomunicaciones hacia la Web 2.0. Adicional a este enfoque, se propone utilizar también la Web para enriquecer los servicios del operador, pero a diferencia de WIMS 2.0, que utiliza un Servidor de Aplicaciones IMS y la API REST para implementar la capa de adaptación entre ambos mundos, en MIDDIS se utiliza la Lógica de Mediación para adaptar las funcionalidades SIP-IMS a las funcionalidades de SOA-WS y viceversa.
Definición de la Arquitectura de una Plataforma de Prestación de Servicios, utilizando conceptos genéricos de IMS y SOA	Definición de la arquitectura de una Plataforma de Prestación de Servicios basada tanto en los conceptos genéricos extraídos de IMS (Telecomunicaciones) como en los de SOA (Internet), ya que por sí solas, como se reconoce en MIDDIS, proveen arquitecturas limitadas de plataformas de servicios que dependen de las tecnologías subyacentes.	En MIDDIS no se define una Plataforma de Prestación de Servicios como tal, sino solamente la arquitectura de una Lógica de Mediación, basada en SOA, para facilitar la interacción de servicios basados tanto en IMS como en SOA. Debido al alcance en la definición de una SDP que maneje la convergencia de las Telecomunicaciones e Internet/IT, y de acuerdo a los requerimientos de MIDDIS, todo lo concerniente a una SDP constituye un elemento teórico fundamental para la definición de la Lógica de Mediación.
Modelo de Comunicaciones basado en SIP para Servicios Web en Tiempo Real	Utilización de WS en tiempo real en el dominio de las Telecomunicaciones. Para ello se proponen y diseñan máquinas de estados mejoradas para el manejo de las llamadas en el RT-WSCM. En MIDDIS se diseñan e implementan máquinas de estados finitos para la manipulación de los protocolos de comunicaciones de las redes subyacentes.	En MIDDIS se aplica el concepto de trabajar con motores asíncronos, basados en eventos, como JAIN SLEE, el cual fue propuesto por (Cheng, y otros, 2008) como trabajo futuro. Esto ratifica el hecho de que la utilización de la especificación de JAIN SLEE, para abordar las necesidades de utilización de WS, con mantenimiento del estado, en un entorno de Telecomunicaciones como IMS, es una opción bastante viable.

2.2.2. Antecedentes para el Manejo de Sesiones en los Servicios Web

Los sistemas telefónicos saben cuándo comienza y termina una llamada. Pero en el mundo SOA, de sistemas débilmente acoplados y de aplicaciones distribuidas, el manejo de las sesiones no es tan simple. Es por ello que, por extraño que pueda parecer, hasta el año 2003 para los WS no existía el concepto de sesión, descrito de forma general como un mecanismo

para correlacionar varios mensajes con el fin de lograr alguna semántica visible de una aplicación (Little, 2005). De este modo, típicamente los WS son sin estado, lo cual significa que cada invocación a un WS debe contener toda la información necesaria para procesar la petición, dado que el procesamiento depende solamente de dichos datos, y por esa razón este diseño simplifica enormemente la implementación de los WS. En consecuencia, la mayoría de las arquitecturas orientadas al servicio del mundo real todavía dependen de protocolos de bajo nivel o de tecnologías propietarias, que pueden restringir la flexibilidad de los servicios y el potencial de la reutilización. (Dornan, 2007)

Para los servicios de valor agregado, las plataformas de prestación de servicios han evolucionado desde la tradicional IN hacia las recientes plataformas de intermediación de servicios basadas en los WS. En estas condiciones, el foco del problema se ha convertido en encontrar cómo proveer WS para comunicaciones en tiempo real. Sin embargo, a pesar de que la adopción de la tecnología de WS en la capa de servicios de la infraestructura de las Telecomunicaciones tendrá un impacto significativo en la evolución del mercado abierto de servicios, la utilización de un modelo de comunicaciones estándar para el desarrollo de los WS de Telecomunicaciones está lejos de ser trivial (Cheng, y otros, 2008). Las técnicas tradicionales como: cookies, reescritura de la URL y la información oculta en los formularios no son las adecuadas para el manejo de las sesiones y transacciones SOAP de los WS. En consecuencia, se ha requerido de la creación y el desarrollo de nuevos mecanismos para manejar las sesiones y las transacciones para los WS, los cuales van desde la utilización de tecnologías externas, casos en los cuales se ha utilizado principalmente a SIP (Dong y Newmarch, 2005), hasta aquellos mecanismos que utilizan nuevos estándares de WS, como WS-Addressing, WS-Coordination y WS-Context, en las que se complementan los encabezados de invocación SOAP con la información de la sesión.

Motivados por la creciente necesidad de la utilización del concepto de sesión en los procesos de gestión de los WS, a continuación se describen algunas de las alternativas que se han desarrollado para resolver esta problemática. Estas son importantes para MIDDIS porque evitan sobrecargar la gestión de sesiones a través de SIP.

2.2.2.1. Gestión de la Sesión y de las Transacciones en los Servicios Web utilizando SIP

En (Dong y Newmarch, 2005) se desarrolla un nuevo mecanismo para dar soporte a la Gestión de la Sesión en los WS utilizando el protocolo SIP, y en base a este mecanismo también se propone una solución simple para la gestión de transacciones para WS.

En las aplicaciones orientadas a la sesión existen dos problemas principales que necesitan ser resueltos. Uno de ellos es cómo gestionar en el lado del servidor múltiples sesiones; el otro de ellos es cómo mantener durante la comunicación el estado entre el cliente y el servidor. En este sentido, en (Dong y Newmarch, 2005) se analizaron dos métodos para la gestión de las sesiones en los mensajes SOAP. El primer método consiste en la integración de la gestión de la sesión en los Servidores Web; este es el caso del software Zap, del proyecto REGNET (Tsenov, 2002), que, aunque proporciona una funcionalidad sencilla para el desarrollador y el administrador del sistema, presenta una clara desventaja al hacer parte del servidor Apache, y, por tanto, no permitir la cooperación con otros servidores Web para la gestión de la sesión. La segunda

solución, provista por Microsoft Corporation, consiste en la utilización del encabezado SOAP para proveer servicios con estado; sin embargo, esta versión de una sesión SOAP se basa en el mecanismo de las cookies, para lo cual el desarrollador tiene que asegurarse con cada petición de que la aplicación cliente trate las nuevas etiquetas SOAP como cookies HTTP y las envíe de regreso al servidor, con lo cual a su vez se limita el transporte de SOAP sobre HTTP, y por tanto la cooperación de SOAP con otros protocolos de transporte.

Para la demostración del mecanismo de gestión de la sesión se implementó un Sistema de WS de Venta de Videos en Línea: un usuario de WS puede establecer una sesión con el sistema (luego de que la sesión se configura, la aplicación cliente obtiene un identificador único de sesión que es almacenado por el servidor SIP, así como también es introducido en el encabezado de cada mensaje SOAP transmitido entre la aplicación cliente y el servidor) y luego invocar los WS del proveedor; luego de que el usuario finaliza todas las actividades de compra, por medio de los WS, puede terminar dicha sesión (el identificador único de la sesión se elimina tanto del servidor SIP como del servidor del WS). Este mecanismo involucra la adición de la información de la sesión en el encabezado SOAP, y la utilización de SIP para el manejo del estado de la sesión de los WS. De esta manera, se utiliza un protocolo estándar para generar el identificador de la sesión, gestionar la sesión para los WS, y para proveer extensibilidad en el tema de la seguridad.

Para la demostración del mecanismo de gestión de transacciones en los WS se implementó un Sistema de WS de Banca electrónica: en él se desarrolló la gestión de la transacción en la transferencia de dinero entre dos bancos; sin embargo, el identificador único de sesión utilizado en el mecanismo de gestión de la sesión no puede ser usado para identificar una transacción entre dos bancos debido a que el cliente puede crear muchas transacciones dentro de una sesión. Por lo tanto, para llevar a cabo la gestión de transacciones en los WS se necesita utilizar un gestor de transacciones, responsable de la coordinación de todas las actividades de los participantes en una transacción. En este mecanismo se sigue utilizando SIP para gestionar el estado de la sesión, y se introduce un gestor de transacciones, el cual es un WS que necesita comunicarse con el servidor SIP para controlar las transacciones entre los participantes.

2.2.2.2. Gestión Confiable de la Sesión en los Servicios Web utilizando SIP

En (Rapeeporn y Benchaphon, 2007) se emplea SIP y algoritmos de anillo para manejar sesiones confiables entre un cliente o solicitante de un servicio y un proveedor WS. SIP se utiliza en los Agentes de Usuario Servidores (User Agent Server, UAS) SIP para manejar una sesión única, por cada solicitud al servicio. La información de la sesión se mantiene en cada UAS y se utiliza entre el solicitante del servicio y los WS. Al mantener la seguridad en la sesión la información de sesión se difunde (multicast) a los diferentes UAS que estén conectados en la red, en forma de anillo virtual. Cuando el UAS principal falla, se utiliza un algoritmo de anillo para elegir el segundo UAS con la mayor prioridad.

La arquitectura del sistema está constituida por cuatro UAS SIP, donde uno de ellos es el servidor líder o primario. A su vez, se tienen WS y Agentes de Usuario Clientes (User Agent Client, UAC) SIP.

Un UAS es un servidor que genera un identificador único de sesión para cada UAC; el formato de un identificador de sesión es: `YYYYMMDDhhmmss@<UAC IP address>`, donde `<UAC IP address>` es la dirección IP del cliente. Cuando un UAC desea invocar un WS, el UAC necesita enviar una petición al UAS primario o líder. Al establecerse la conexión, el UAS líder mantiene y monitorea la sesión entre el UAC y el WS por medio de la utilización de tres módulos: el manejador de peticiones, el servicio de multidifusión y el sistema de monitoreo. El UAS se comunica con el UAC a través de mensajes SIP. Un UAC es una aplicación cliente que permite a los usuarios invocar un servicio por medio del envío de mensajes de petición hacia el UAS.

En la arquitectura del modelo propuesto se utilizan WS genéricos que dan soporte a la interacción entre clientes u otros WS, de una manera interoperable, a través de la red. La interfaz de un WS es descrita por la WSDL y el WS se comunica con quien lo solicita o con el UAC por medio de mensajes SOAP. Antes de que un WS permita a un UAC la invocación de sus servicios, tiene que comprobar el identificador de la sesión con el UAS SIP.

En la implementación, y para demostrar el modelo propuesto en (Rapeeporn y Benchaphon, 2007), se utiliza el ejemplo de un sitio Web de comercio electrónico que incluye: el UAC, que es una tienda de compras, el WS 1 que es un proveedor de WS, y el WS 2, que es un WS de envíos. Los mecanismos descritos son: monitoreo, multidifusión, inicio de sesión, comunicación de los servicios y terminación de la sesión.

2.2.2.3. WS-Addressing

Estrictamente hablando, la gestión de las sesiones no ha estado del todo ausente de SOA. Por ejemplo, con WS-Addressing (Box, y otros, 2004) se ha incluido el soporte a la sesión como un elemento opcional. Este es un estándar maduro de la W3C, utilizado para enrutar los mensajes SOAP.

Este es un estándar de servicios Web importante para el direccionamiento del servicio en una vía. Define dos estructuras interoperables que conducen información, típicamente proporcionada por protocolos de transporte y sistemas de mensajería. Estas estructuras normalizan la información subyacente y la convierten en un formato uniforme, que puede ser procesado independientemente del transporte o de la aplicación. Las dos estructuras son: la Referenciadel PuntoFinal (Endpoint Reference, EPR) y los Encabezados de Información de los Mensajes. Un punto final de un WS es una entidad, procesador, o recurso al cual puede hacerse referencia y hacia donde pueden dirigirse los mensajes de los WS. Las EPR pueden conducir la información necesaria para identificar/referenciar un punto final de un WS, con una resolución mejorada que va más allá de las direcciones URI o IP tradicionales, y pueden utilizarse en variadas y diferentes formas: las referencias de los puntos finales son adecuadas para la conducción de la información necesaria para acceder al punto extremo de un WS, pero también se utilizan para proporcionar direcciones a los mensajes individuales enviados hacia y desde los WS. Para tratar este último caso de uso, esta especificación define una familia de encabezados que permiten el direccionamiento uniforme de mensajes, independientemente del transporte subyacente. Estos encabezados conducen las características del mensaje extremo a extremo, incluyendo el direccionamiento para los puntos finales, tanto de la fuente como del destino, así como la identidad del mensaje.

Ambas estructuras están diseñadas para ser extensibles y reutilizables, por lo que otras especificaciones pueden construir y hacer uso de las EPR y de los encabezados de los mensajes. No obstante, WS-Addressing condiciona las sesiones a direcciones SOAP particulares y funciona solamente para las SOA que utilicen WS-Addressing. Es decir, que el modelo de sesión de WS-Addressing provee el acoplamiento entre la información del punto extremo del WS y los datos de sesión, lo cual es análogo a las referencias a objetos en sistemas de objetos distribuidos. Pero las SOA basadas en WS a menudo no lo hacen, ya que en lugar de ello pueden ser localizadas a través de URL, y por otra parte, el direccionamiento se necesita realmente sólo para los WS que a traviesan transportes que no son HTTP (Dornan, 2007).

2.2.2.4. WS-Coordination

El grupo WS-TX (Web Services Transaction), que se encuentra dentro de Oasis, estaba encargado de desarrollar 3 estándares: WS-Coordination (Oasis, 2009a), WS-AtomicTransaction (Oasis, 2009b) y WS-BusinessActivity (Oasis, 2009c), los cuales se aprobaron en Abril del 2007. WS-Coordination (Dornan, 2007) es el estándar base, dirigido hacia la coordinación de metadatos. En lugar de especificar cómo se pueden coordinar los WS, este estándar les proporciona un marco para la transferencia de información acerca de la coordinación. En la parte superior de WS-Coordination, y estratificados, hay dos estándares que dirigen la coordinación en sí: WS-AtomicTransaction y WS-BusinessActivity. El primero de los estándares dirige los servicios punto a punto, y relativamente simples, y el último trata con aplicaciones más complejas.

Esta especificación describe un marco de trabajo para un servicio de coordinación (o coordinador), el cual consiste en los siguientes servicios:

- Activación: con una operación que le permite a una aplicación crear una instancia de coordinación o contexto.
- Registro: con una operación que le permite a una aplicación registrarse para el uso de los protocolos de coordinación.
- Un conjunto de protocolos de coordinación de tipo específico.

Las aplicaciones utilizan el servicio de activación para crear el contexto de coordinación para una actividad. Una vez que una aplicación adquiere el contexto de coordinación éste se envía, a través de cualquier medio apropiado, hacia otra aplicación. El contexto tiene la información necesaria para registrarse en la actividad, especificando el comportamiento de la coordinación que la aplicación seguirá. Adicionalmente, una aplicación que reciba un contexto de coordinación puede utilizar el servicio de registro de la aplicación original, o puede utilizar uno de los que especifique, por interposición, un coordinador de confianza. De esta manera, una colección arbitraria de WS puede coordinar su operación de acoplamiento.

Por sí mismo, este estándar no define todas las características requeridas para una solución integral. WS-Coordination es un bloque de construcción que se utiliza en conjunto con otras especificaciones y protocolos específicos de aplicación para adaptarse a una amplia variedad de protocolos relacionados con las operaciones de WS distribuidos.

Con WS-Coordination los WS tienen que manejar ellos mismos todos los contextos, por lo que, en lo concerniente a las sesiones, probablemente esto limitará a WS-Coordination a aplicaciones relativamente simples. Además, los estándares del WS-TX (Oasis, 2009d) también están limitados a trabajar con SOAP.

2.2.2.5. WS-Context

WS-Context (Dornan, 2007) (Oasis, 2007) (Oh, y otros, 2006) define un marco de trabajo abierto para dar soporte a la composición, coordinada y transaccional (con estado), de múltiples aplicaciones de WS. Fue creado por el Marco de Trabajo para Aplicaciones Compuestas de Servicios Web (Web Services Composite Application Framework, WS-CAF), que también se encuentra dentro de Oasis, y que originalmente se encargaría de producir 2 estándares más. Estos últimos manejarían la coordinación de múltiples WS en aplicaciones compuestas, y la gestión de transacciones individuales, pero ambos fueron abandonados. WS-Context fue aprobada en Abril del 2007, al mismo tiempo que los estándares del WS-TX.

Su funcionalidad es manejada en parte por el Lenguaje de Ejecución de Procesos de Negocio (Business Process Execution Language, BPEL) y en parte por WS-TX, una especificación, que es su rival, y es desarrollada dentro de Oasis por un grupo que incluye a IBM y a Microsoft.

Este estándar está orientado a la gestión flexible de sesiones para las SOA, permitiendo que los servicios mantengan el estado sin depender de URL o protocolos de bajo nivel como TCP/IP; en otras palabras, tiene como objetivo ser para SOA lo que han sido las cookies para la Web. Para ello, define una estructura XML en la cual se almacenan los elementos de datos relacionados con el entorno de ejecución de servicios. En dicha estructura se estandariza el formato y la disposición de los datos de seguimiento de las sesiones, dándole a cada mensaje SOAP un elemento `Context` dentro de su encabezado. Entre las etiquetas de `Context` está un `Context-Identifier` que contiene un identificador único para la sesión, similar a una cookie. Los elementos y atributos opcionales del contexto especifican la autoridad que generó el identificador de la sesión, por cuanto tiempo es válido, y cualquier dato relacionado con seguridad. Los contextos pueden pasarse por valor (toda la información requerida para utilizar el contexto está presente en la estructura de datos) o puede pasarse por referencia (sólo un subconjunto de la información está presente en la estructura de datos y el resto debe ser obtenida por el servicio receptor).

Los participantes en una SOA pueden generar sus propios identificadores de sesión, pero en un sistema distribuido grande puede ser más conveniente centralizar el proceso. WS-Context hace esto posible a través del Gestor de Contexto, un WS opcional que puede fijar y editar los contenidos de los elementos del contexto, así como también da soporte al paso de contexto por referencia. Al estar construido por encima de la WSDL, esto también simplifica la administración, ya que, por requerimiento, el Gestor de Contexto sabe acerca de cada sesión dentro de la SOA.

Aunque la intención de WS-Context es ser incorporada dentro de los encabezados SOAP, ya se ha extendido a otras aplicaciones. Los mismos elementos se pueden utilizar en WS que utilicen, por ejemplo, REST y también se pueden utilizar dentro de las Arquitecturas de

Componentes de Servicios (Service Component Architecture, SCA), que también requiere de sesiones persistentes para la comunicación entre los componentes de las aplicaciones.

2.2.2.6. Resumen comparativo entre los antecedentes para el Manejo de Sesiones en los Servicios Web y MIDDIS

En la Tabla 3 se realiza una comparación entre los antecedentes presentados para el soporte a las sesiones en los WS y MIDDIS.

Tabla 3. Resumen comparativo entre los antecedentes para el Manejo de Sesiones en los Servicios Web y MIDDIS

Proyecto o estándar	Elementos Considerados en MIDDIS	Elementos diferenciadores con MIDDIS
Gestión de la Sesión y de las Transacciones en los Servicios Web utilizando SIP	Utilización de SIP como un soporte adicional a los procesos de gestión de las sesiones y el manejo del estado en los WS, debido a que la información de contexto que maneja el dominio IMS es relevante para el dominio SOA, específicamente para los WS.	En MIDDIS solamente se utilizan los fundamentos y resultados conceptuales de (Dong y Newmarch, 2005).
Gestión Confiable de la Sesión en los Servicios Web utilizando SIP	Utilización de SIP para el manejo de sesiones en el entorno IMS.	En MIDDIS se utilizan parcialmente los fundamentos y resultados tanto conceptuales como prácticos de (Rapeeporn y Benchaphon, 2007), ya que el manejo de las sesiones para servicios convergentes IMS/SOA se plantea a través de la interacción SIP/WS, y no solamente a través de SIP.
WS-Addressing		No se considera su utilización en MIDDIS por las limitaciones en la forma de localización de los servicios, la cual es realizada en este estándar a través de referencias fijas.
WS-Coordination		No se considera su utilización en MIDDIS por cuanto el alcance y la complejidad de esta última solamente se limita al manejo de sesiones enlazadas a un servicio o una actividad y no a sesiones basadas en transacciones.
WS-Context	Se propone la utilización de este estándar por sus características diferenciadoras con respecto a los anteriores estándares de WS, y por sus ventajas para el manejo de las sesiones en entornos de WS: - Diseñado específicamente para la gestión de las sesiones. - Es extensible a otras aplicaciones que no necesariamente se basen en SOAP.	

Capítulo 3

LÓGICA DE MEDIACIÓN PARA LA INTERACCIÓN DE SERVICIOS BASADOS EN SOA E IMS – MIDDIS

3.1. Modelo del Ambiente

Actualmente las telecomunicaciones requieren de una plataforma de servicios que maneje la convergencia Telecomunicaciones/Internet/IT. Por ahora, como habilitadores para la convergencia las telecomunicaciones utilizan a IMS y la comunidad de Internet/IT a SOA. En este sentido: i) IMS utiliza SIP como protocolo de señalización para llamadas y sesiones, y define entidades funcionales y puntos de referencia tanto para las redes de telecomunicaciones fijas como para las móviles. Sin embargo, IMS carece de la arquitectura de una plataforma de servicios completa. ii) Por su parte, SOA es la arquitectura de una plataforma de servicios, que abstrae diversas funciones del sistema en servicios e integra los servicios en las aplicaciones. (Rolan y Hu, 2008)

Teniendo en cuenta las propuestas para la convergencia IMS/SOA presentadas en 2.2, en la Figura 1 se presenta el Modelo del Ambiente (Serrano, 2005) de MIDDIS.

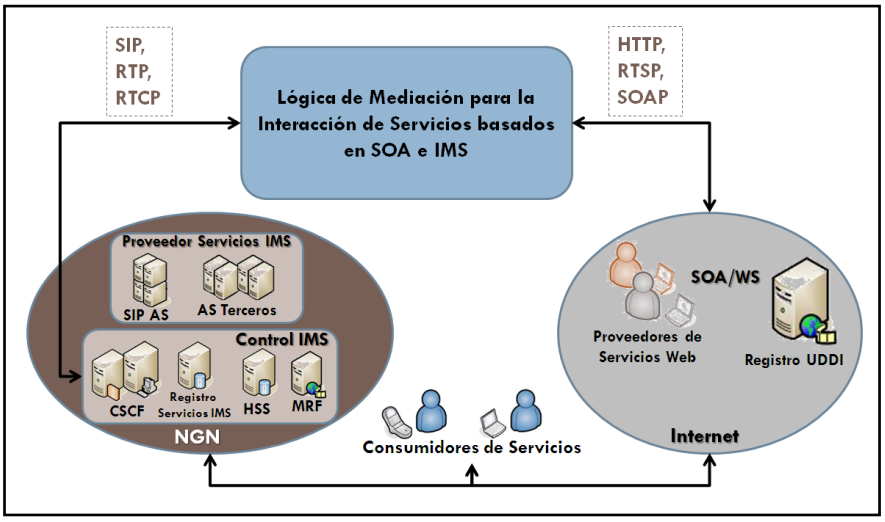


Figura 1. Modelo del Ambiente del Sistema

En ella se ilustra la integración de estas dos plataformas para extender su rango de aplicabilidad. La característica esencial de la lógica de mediación consiste en que IMS utilizará a SOA para integrar sus propios elementos software con componentes externos y de esta manera lograr la combinación de las facilidades de la Web y de IMS para exponer un conjunto de servicios enriquecidos para ambos mundos. La mayoría de piezas significativas de este rompecabezas de interacción e integración son las relaciones entre los servicios y entre el operador de red y las terceras partes, los cuales obtienen acceso y características de la red a

través de programas ofrecidos por el operador, o suministran al operador de red su propia información o características.

Otras de las características básicas de MIDDIS son la provisión de la adaptación entre SIP y la pila de protocolos de los WS, y para ello extiende sus funcionalidades y capacidades; la provisión de la adaptación de los Registros de los servicios IMS a Registros únicos de Servicios Web, y con ello se unifica el acceso a la información que describe los servicios; y adicionalmente la propuesta de adaptación de los medios de Internet (tipos de medios Internet: RTSP, HTTP, etc.) a los medios de IMS (RTP y RTCP para el transporte del flujo IP multimedia en el plano del usuario), para asegurar la interoperación entre el plano de medios de IMS/SIP y el de SOA/WS. De esta manera, la lógica de mediación permite que clientes con capacidades únicas IMS tengan acceso a los servicios prestados por las SOA/WS o por el contrario que clientes de las SOA/WS accedan a los servicios prestados por IMS, y con esto a su vez se logra el acceso a servicios IMS/SOA convergentes.

3.1.1. Descripción de características

- MIDDIS da soporte al mundo IMS/SIP, por lo que los clientes de servicios IMS, dominio de las telecomunicaciones, que acceden a ella son usuarios SIP-3GPP, con capacidades de transporte de datos IP a través de RTP y RTCP.
- MIDDIS también da soporte al mundo SOA, basándose en los WS para los procesos de creación de servicios convergentes IMS/SOA, por lo cual da soporte a la arquitectura de WS los cuales son considerados como una tecnología de mediación, cuya filosofía en el desarrollo de aplicaciones permite compartir e integrar recursos.
- La definición e implementación de MIDDIS para la comunicación entre el mundo IMS/SIP de las telecomunicaciones y el mundo SOA/WS de las IT, tiene en cuenta, por una parte, tanto las características de la arquitectura de servicios IMS como las características de las Arquitecturas Orientadas a Servicios, y particularmente los roles que existen en la arquitectura de los WS, y, por otra parte, considera también las limitaciones de ambos mundos para la creación de servicios convergentes. La limitación más importante en este proceso de convergencia consiste en que los WS presentan como arquitectura una pila de capas en la que todos sus elementos están basados en XML, sin embargo SIP no se basa en XML, por lo cual actualmente las aplicaciones de WS no soportan SIP, y viceversa. Por esta razón, la definición e implementación de MIDDIS depende de las características en común y diferenciadoras de las plataformas IMS y SOA, lo cual a su vez determina el grado de integración que se requiere para su obtención.
- MIDDIS da soporte a la interacción entre plataformas IMS, basadas en SIP, y plataformas SOA, específicamente de WS, creando una lógica de mediación para el soporte de la gestión de sesiones SIP/WS y el control de servicios convergentes IMS/SOA .
- Proporciona de manera fundamental la característica de exposición de capacidades, tales como la gestión de la sesión, establecimiento de medios, el registro y acceso a servicios, hacia ambos mundos, ya través de pasarelas basadas tanto en SIP como en WS, que permiten y facilitan la interacción bidireccional entre clientes y proveedores de SOA-WS y usuarios y servicios del dominio IMS. Este escenario, para el desarrollo de servicios convergentes, se puede ver como una red global compuesta de diversas redes interconectadas a través de pasarelas, a nivel de la capa de aplicación y control, que

funcionan en base a una lógica de ejecución. Esta última en conjunto con las pasarelas conforman la lógica de mediación para la interacción entre servicios basados en IMS y SOA.

La Tabla 4 muestra el resumen de las características de alto nivel de la lógica de mediación para la interacción de servicios basados en IMS y SOA, que sirvieron de base para la identificación de sus requerimientos funcionales y no funcionales.

Tabla 4. Caracterización de la Lógica de Mediación

Característica	Descripción
Protocolo de control de la sesión en IMS	SIP-3GPP
Protocolo de acceso a la plataforma IMS	SIP-3GPP
Protocolo para el transporte de medios en IMS	RTP, RTCP
Control de la sesión en IMS	CSCF
Servidores de aplicaciones en IMS	AS IMS(SIP), AS JAIN SLEE (Lógica de Mediación)
Registro de Servicios IMS	Registro Servicios IMS
Consumidores de Servicios IMS	Clientes SIP(3GPP)-IMS, Clientes Web
Protocolo para el acceso a Servicios Web	SOAP sobre HTTP
Lenguaje para la descripción de los WS	WSDL
Protocolo para el transporte de medios Internet	HTTP, RTSP
Servidores de Servicios Web	Proveedor de Servicios Web
Registro de Servicios Web	UDDI
Consumidores de Servicios Web	Clientes SIP(3GPP)-IMS, Clientes Web
Gestión de Sesiones SIP/WS	Lógica de Mediación
Control de Servicios Convergentes IMS/SOA	Lógica de Mediación
Pasarela SIP/WS (Gestión de la sesión SIP/WS)	Adaptador de WS a SIP, Adaptador de SIP a WS
Gestión del Registro de Servicios IMS/WS	Publicación, Actualización y Consulta del Registro (basado en UDDI) de Servicios IMS y de WS
Pasarela de Medios IMS/WS	Adaptador de Medios WS a Medios IMS

3.1.2. Requisitos de la solución

A partir de las características mostradas en la Tabla 4, se identificaron los siguientes requisitos para la Lógica de Mediación.

3.1.2.1. Funcionales

- Dar soporte a la señalización que se lleva a cabo en IMS a través del SIP-3GPP. De esta forma se provee el acceso a la plataforma IMS desde cualquier elemento externo a ella.
- Dar soporte a los protocolos RTP/RTCP a través de los cuales en IMS se lleva a cabo el transporte de los medios de la comunicación.
- Realizar una comunicación bidireccional con el Núcleo de IMS, con el fin de dar soporte a la señalización y control en IMS.
- Recibir, analizar y ejecutar las peticiones (de registro de servicios, ejecución de interacción de servicios, gestión de servicios convergentes, etc.) provenientes desde los Proveedores

de Servicios IMS, las cuales son recibidas inicialmente por el CSCF, y posteriormente enviadas hacia la Lógica de Mediación.
- Permitir el ingreso de la información de los servicios IMS siempre que sea requerido un proceso de registro de servicios en la red.
- Permitir el acceso de Clientes IMS a servicios convergentes basados en IMS y SOA.
- Dar soporte al protocolo SOAP, necesario para el transporte de los mensajes de información en los WS. Con ello también se asegura el manejo de otros procesos de los WS basados en la WSDL, por ejemplo.
- Dar soporte a protocolos de transporte de medios en Internet, tales como HTTP, con lo cual también se brinda el manejo de elementos como el Registro de Servicios Web.
- Recibir, analizar y ejecutar las peticiones provenientes desde los Proveedores de Servicios Web.
- Dar soporte a las funcionalidades requeridas por el Modelo de Registro de los WS.
- Permitir el acceso de Clientes Web a servicios convergentes basados en IMS y SOA.
- Realizar la adaptación de la señalización SIP a la pila de protocolos de los WS y viceversa, con el fin de llevar a cabo la interacción a nivel de control de servicios y de señalización, entre IMS/SIP y SOA/WS. Es decir, permitir tanto la gestión de sesiones SIP/WS como el control de servicios convergentes IMS/SOA.
- Realizar la publicación, actualización y búsqueda de los Registros de los Servicios IMS Y WS, con el fin de unificar el proceso de gestión del registro de servicios, y por consiguiente llevar a cabo la unificación de las descripciones de servicios tanto IMS como Web.
- Realizar la adaptación de los Medios de los WS a los Medios IMS, con el fin de llevar a cabo la interacción de los datos, del plano del usuario, entre IMS/SIP y SOA/WS.

3.1.2.2. No funcionales

- Utilizar un entorno de ejecución de Lógica de Servicios de Comunicaciones con el fin de proporcionar las características que los servicios de telecomunicaciones requieren, tales como: invocaciones comúnmente asíncronas, eventos de grano fino y de alta frecuencia, componentes livianos de rápida creación y eliminación, múltiples fuentes de datos, transacciones livianas, despliegue distribuido a lo largo de la red, alta disponibilidad (de 3 a 5 nueves), etc.
- Extender la pila básica de los WS mediante la utilización de nuevas tecnologías que actualmente se están estandarizando y que se orientan cada vez más a WS con estado y al manejo de sesiones.

3.2. Arquitectura de Referencia

Algunas veces IMS y SOA se ven como tecnologías competitivas, pero en realidad, casi todas las implementaciones de IMS utilizarán SOA, y para los proveedores de servicios muchas aplicaciones SOA también incluirán a IMS (Rolan y Hu, 2008). Por otra parte, en el área de creación de ecosistemas de servicios la oportunidad de un servicio dado puede ser visualizada como un ecosistema SOA, un ecosistema IMS, o una combinación de los dos (Nolle, 2008). De acuerdo a lo anterior, la propuesta de MIDDIS, para la creación de un entorno de mediación para la interacción de servicios basados en IMS y SOA, se orienta a que los servicios se

visualizarán como una combinación de los ecosistemas SOA e IMS. Para que esto se lleve a cabo SOA va a ser utilizado en la capa de aplicación de IMS, ver Figura 2, y el entorno de mediación se construirá alrededor de la integración entre los elementos funcionales de IMS y los principios de SOA en el desarrollo de aplicaciones.

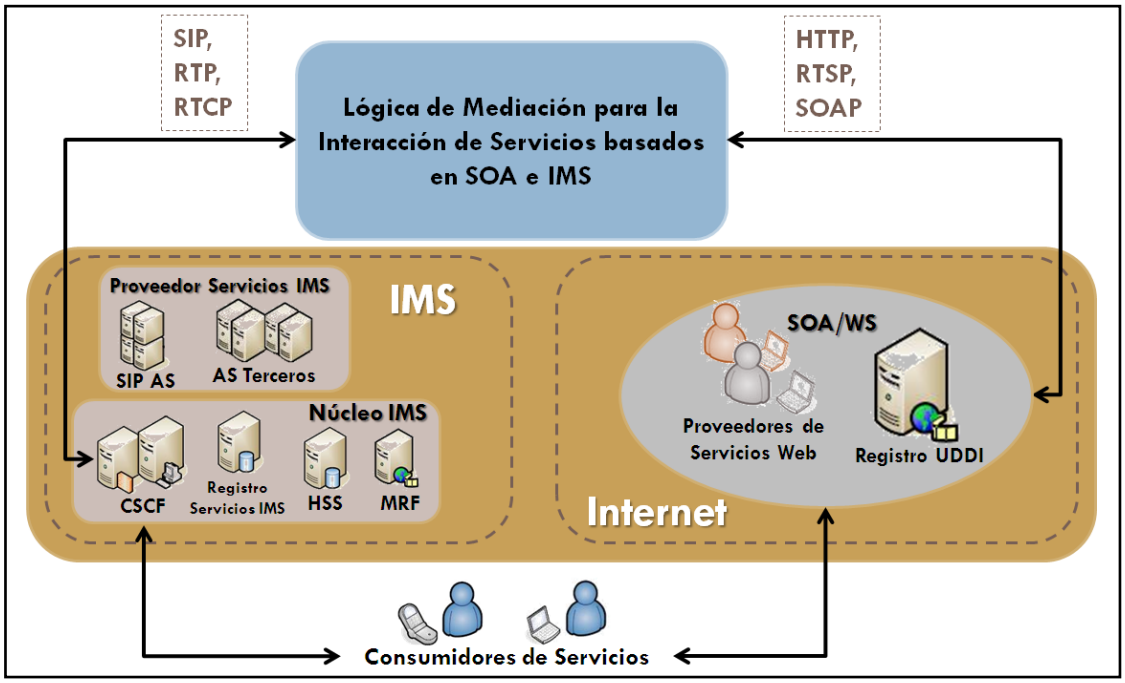


Figura 2. Modelo para la Interacción de servicios basados en IMS/SOA

Para satisfacer los requisitos estipulados en la sección 3.1.2, en la Figura 3 se plantea la arquitectura de referencia para la solución de la Lógica de Mediación para la Interacción de Servicios basados en SOA e IMS. MIDDIS se encuentra dividida en cinco subsistemas funcionales: i) Subsistema de Mediación SIP/SOAP (SSMIDD), ii) Subsistema de Registro de Servicios IMS/WS (SSREGS), iii) Subsistema de Medios IMS/WS (SSMED), iv) Subsistema de Transmisión de Eventos (SSTE), y v) Subsistema de Interfaces de Recursos de Red(SSIRR). Los colores usados en la Figura X indican lo siguiente:

- *Azul oscuro*: Representa la Lógica de Mediación para la Interacción de Servicios basados en SOA e IMS.
- *Azul claro*: Representa las entidades externas a la Lógica de Mediación sobre las cuales se tiene control y/o se realizan aportes, ya que entran en contacto directo con ésta. Este grupo de entidades está conformado por: el Componente de provisión de servicios IMS, el Cliente IMS, el Componente SOA/WS, y el Cliente Web.
- *Naranja*: Representa las entidades externas a la Lógica de Mediación sobre las cuales no se tiene control. En este caso se encuentra el Núcleo de IMS.

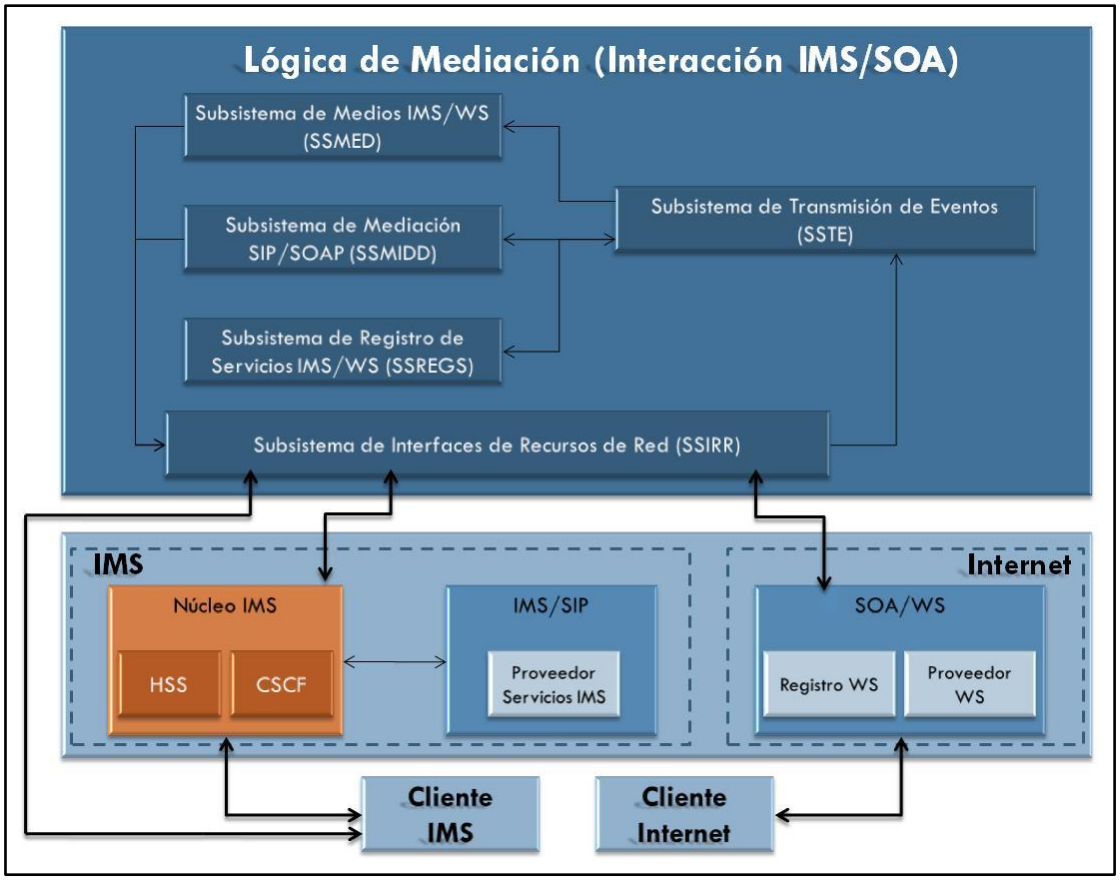


Figura 3. Arquitectura de Referencia

A continuación se describe en detalle cada uno de los subsistemas que conforman la Lógica de Mediación para la Interacción de Servicios basados en SOA e IMS.

3.2.1. Descripción de la Arquitectura de Referencia del Sistema

3.2.1.1. Diagrama de Subsistemas

En la Figura 4 se muestra el Diagrama de Subsistemas, en el cual se indican las interfaces de comunicación con las entidades externas a la Lógica de Mediación y entre subsistemas. En el diagrama se puede observar que el SSIRR se comunica con el Núcleo de IMS a través de la señalización SIP y con el cliente IMS a través de SIP y RTP/RTCP. Así mismo, este subsistema provee interfaces HTTP, SOAP, WSDL, y UDDI hacia el Registro de WS, e interfaces HTTP y SOAP hacia el Proveedor de WS, los cuales son elementos que pertenecen específicamente al componente SOA/WS. Adicionalmente, el SSIRR se comunica con el SSTE cada vez que ocurre un evento proveniente de las interfaces de los recursos de la red subyacente. Dicho evento es enviado por el SSTE a cualquiera de los subsistemas restantes que haya registrado un interés en él.

De manera inversa, el SSMIDD y el SSMED por una parte pueden enviar al SSIRR los resultados de los procesos de mediación, que deban ser transmitidos a las entidades externas a la Lógica de Mediación, y por otra parte el SSMIDD y el de SSREGS también generan resultados parciales, los cuales pueden ser enviados en forma de eventos hacia los subsistemas restantes, y a través

del SSTE, en aquellos casos en los que es necesario un procesamiento adicional, por parte de otro subsistema, antes del envío de los resultados finales de mediación hacia el SSIRR.

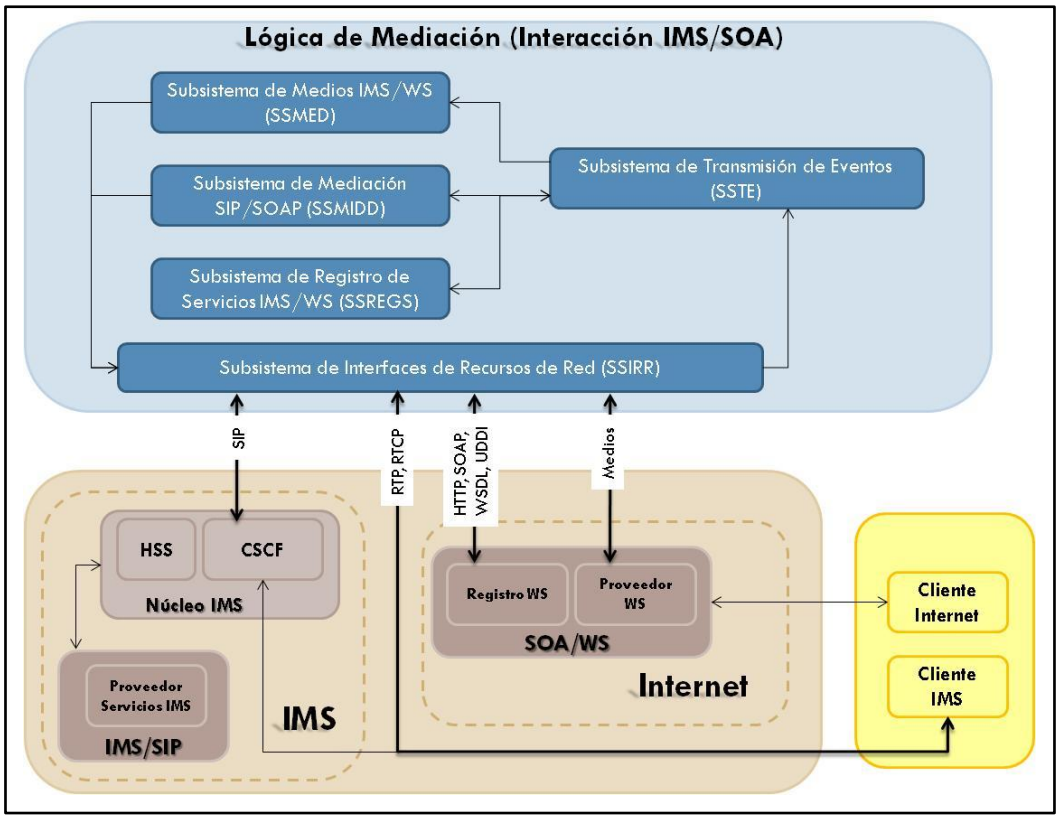


Figura 4. Diagrama de Subsistemas

3.2.1.1.1. Subsistema de Interfaces de Recursos de Red(SSIRR)

Conecta la Lógica de Mediación y la infraestructura de red subyacente constituida por sistemas tales como dispositivos de red, pilas de protocolos, bases de datos, etc. En este caso, la red subyacente corresponde específicamente a los elementos del Núcleo de IMS y al Cliente IMS, por una parte, y al componente SOA/WS de provisión de Servicios Web, ubicado en la capa de aplicación de IMS, por otra. Teniendo en cuenta lo anterior, este subsistema provee las interfaces adecuadas para gestionar (suscribir y arbitrar) los eventos provenientes de cada uno de los recursos que estos elementos manejan, por lo cual provee interfaces para los protocolos SIP y RTP/RTCP, para el Núcleo de IMS y el Cliente IMS respectivamente, e interfaces para los protocolos HTTP, SOAP, WSDL, UDDI y Multimedia, para el componente SOA/WS de Internet.

Al recibir los eventos este subsistema los adapta para convertir los protocolos y eventos específicos de la red en eventos genéricos (entiéndase como eventos representados por medio de un lenguaje de programación genérico), cuyo significado sea equivalente, y además de ello se encarga de gestionar los contextos de los flujos de eventos relacionados con el fin de proporcionar un medio común para que los elementos (red subyacente, Subsistemas de Mediación) que emiten y reciben dichos eventos compartan algunos de sus atributos; finalmente, este subsistema se encarga de enviar los eventos adaptados al SSTE para su direccionamiento hacia el Subsistema de Mediación interesado en ellos. En sentido inverso, este Subsistema recibe y maneja las solicitudes hechas por los Subsistemas encargados de los

procesos de Mediación (SSMIDD, SSMED, y SSREGS) con el fin de generar respuestas hacia la red subyacente, para lo cual adapta esos eventos genéricos en eventos específicos del recurso de red al cual se dirigen los resultados de la mediación.

Con este Subsistema se logra que cualquier aplicación esté desacoplada lógicamente de la red subyacente, de manera que las aplicaciones puedan ejecutarse en cualquier red.

3.2.1.1.2. Subsistema de Transmisión de Eventos(SSTE)

Por una parte, asegura y gestiona la correcta transmisión de los eventos enviados por el SSIRR, direccionándolos al Subsistema de Mediación interesado en ellos. Por otra parte, también asegura y gestiona la correcta transmisión de eventos entre el SSMIDD y el SSREGS, en aquellos casos en que se requiera de un procesamiento adicional. Además, verifica el contexto en el cual se realiza la transacción de cada evento, y asocia, si aún no lo ha realizado, el Subsistema interesado en dicho tipo de evento a su contexto, para que pueda tanto recibir como enviar solicitudes dentro del mismo flujo de eventos relacionados y a su vez pueda compartir ciertos atributos con los demás subsistemas que se asocien a ese flujo. En este sentido, el SSTE es el corazón del direccionamiento de eventos (solicitudes de algún protocolo en particular, o simplemente eventos de alguna tecnología de comunicación) de la Lógica de Mediación, permitiendo que los Subsistemas de Mediación se comuniquen entre sí, y con la infraestructura de recursos subyacente. Por lo tanto, este Subsistema posee las funciones necesarias para asegurar la correcta transmisión de cada evento desde su productor hacia su consumidor.

3.2.1.1.3. Subsistema de Mediación SIP/SOAP (SSMIDD)

Para realizar la interoperación entre servicios basados en SOA-WS e IMS, teniendo en cuenta que la limitación más importante en este proceso consiste en que los WS no utilizan SIP, se diseña la interoperación a nivel de señalización y control entre SIP y los WS por medio del SSMIDD. Este subsistema juega el papel de un mediador y un controlador que ejecuta la adaptación entre el plano de señalización de IMS y la Web, y proporciona el control de los servicios convergentes IMS/SOA. Por lo tanto, contiene las funciones de mapeo requeridas entre el plano de control y señalización de la red IMS y los Servidores de Aplicaciones de Servicios Web, además de algunas funciones para la mediación en la gestión del registro de servicios IMS y WS.

Gestión de la Sesión y Control de Servicios basado en SIP/WS

De acuerdo con los Trabajos Relacionados presentados previamente, para lograr la Gestión de la Sesión y el Control de servicios basado en SIP/WS se selecciona la implementación del enfoque utilizado por las aplicaciones tradicionales de VoIP: utilizar a SIP como protocolo de señalización para localizar los servidores e iniciar sesiones y luego utilizar a SOAP en esas sesiones para las invocaciones normales a WS. El principal inconveniente de este enfoque es que se tiene que hacer que los WS reconozcan a SIP (SIP-aware Web Services) (“sippifying applications”), pero este no es un inconveniente ya que la Lógica de Mediación es la encargada de realizar las adaptaciones entre las peticiones SIP y SOAP. De esta manera, una petición SIP,

generada desde un servicio IMS, es adaptada a una petición SOAP, de acceso a WS, y viceversa, para ser interpretada y ejecutada en base a la pila de protocolos propia del nodo destino.

A pesar de que el protocolo SIP carece de mecanismos de control de servicios, ya que en su lógica solo comprende la gestión de sesiones, existen diversos mecanismos mediante los cuales los sistemas de comunicación multimedia basados en SIP pueden intercambiar mensajes de control de servicios (Almeida, 2007). Uno de estos mecanismos se basa en el uso de la mensajería SIP, y de esta manera al emplear mecanismos propios de SIP para el intercambio de mensajes, se puede encapsular y distribuir en éstos los mensajes de control del servicio. Los mensajes SIP más apropiados para esta funcionalidad son la petición `MESSAGE` y la primitiva INFO para el intercambio de señalización durante una sesión. De este modo, para la invocación de WS desde IMS, se selecciona la utilización de SIP como el protocolo de transporte, no solo de los mensajes de control de invocación a WS, sino también de los resultados de dichas invocaciones. Por lo tanto, los mensajes de invocación se transportan en la carga útil de SIP, utilizando los mecanismos de enrutamiento y direccionamiento de SIP, de la misma manera que se está utilizando en las aplicaciones de Mensajería Instantánea, pero en este caso para los procesos de control de los servicios convergentes IMS/SOA.

En MIDDIS, específicamente, se opta por la extensión del uso de la petición estandarizada de SIP del tipo `MESSAGE`, que inicialmente se definió para mensajería instantánea. La característica fundamental en este escenario de convergencia IMS/SOA es que las invocaciones a WS se realiza dentro de sesiones SIP/WS establecidas. No obstante, este enfoque puede sugerir una serie de inconvenientes, donde el más importante es el hecho de que el mapeo de SOAP sobre SIP no está estandarizado lo cual implica que las infraestructuras actuales de WS no podrían utilizarlo, sin embargo para MIDDIS esto no es un inconveniente ya que la Lógica de Mediación es la encargada de la interacción SIP-SOAP.

Por otra parte, y de acuerdo a los trabajos relacionados presentados previamente, para proporcionar un paradigma de comunicaciones basado en WS con características típicas de mediadores (transacciones relacionadas o no con manejo de sesiones, envío fiable de mensajes, coordinación de los intercambios de información, gestión de políticas, etc.), y con capacidades para interactuar eficazmente con el protocolo de señalización SIP de IMS, MIDDIS propone la extensión de la pila básica de protocolos de los WS acogiendo el nuevo estándar WS-Context, para adicionar la capacidad de manejo de sesiones en el dominio de los WS, y con ello mejorar la tarea del SSMIDD.

El mecanismo de control de servicios convergentes IMS/SOA-WS de MIDDIS, cuando se requiere de la creación y ejecución de los mismos, realiza la mediación entre el control de los Servicios IMS, basado en el modelo de mensajes SIP, y el control de Servicios basados en SOA, a través de la arquitectura de WS. Por lo tanto, el control de los servicios en cada entorno es llevado a cabo a través de sus propios métodos y tecnologías.

3.2.1.1.4. Subsistema de Registro de Servicios IMS/WS (SSREGS)

Por una parte, en SOA el registro de servicios da soporte a los procesos de publicación y el descubrimiento de los mismos. Específicamente para los WS se habla del Registro Universal para la Descripción, el Descubrimiento y la Integración, el cual es un registro donde se

almacena la información referente a los servicios. Por otra parte, en las redes SIP/IMS la publicación y el descubrimiento de servicios se basa en un registro especializado basado en el protocolo SIP. Por lo tanto, para lograr la interacción tanto de servicios basados en SOA como en IMS, a través de este subsistema se lleva a cabo la integración de los mecanismos de registro de servicios que existen en ambos entornos, y se llega a un solo mecanismo de registro basado en el modelo de registro de los WS. Con esto, tanto proveedores como consumidores podrán tener acceso a los procesos de publicación así como de descubrimiento de servicios IMS y de servicios basados en SOA.

Entre algunas de las funciones que tiene este subsistema se encuentran: la recepción de eventos para la publicación de las descripciones de los servicios IMS y de los WS; y la búsqueda y retorno de las descripciones de los servicios consultados y/o actualizados.

3.2.1.1.5. Subsistema de Medios IMS/WS (SSMED)

Muchos de los formatos de datos y tipos de medios Web (Protocolo de Transmisión en Tiempo Real (Real-Time Streaming Protocol, RTSP), HTTP, incluyendo la descarga de archivos) no están soportados actualmente por las Pasarelas de Medios o por las Funciones de Recursos Multimedia de IMS. Además, los UA SIP, tanto clientes como servidores, tampoco tienen la capacidad de tratar con muchos de ellos. Por lo tanto, cuando el intercambio extremo a extremo de los datos de una aplicación requiera de la interacción tanto de servicios basados en IMS como de servicios basados en SOA, específicamente en WS, esto no podrá llevarse a cabo. En consecuencia, para asegurar la interacción entre el plano de medios de IMS/SIP y el de SOA/WS, el principal propósito del SSMED es hacer posible que se recupere y reciba el contenido Web, proveniente de los AS, se adapte a los formatos y protocolos de medios de la red IMS, y se envíe hacia el Cliente IMS. El proceso inverso no es imprescindible ya que el entorno Web es mucho más amplio en el soporte a formatos de datos y protocolos.

3.2.1.2. Definición de las Interfaces de los Subsistemas

3.2.1.2.1. Interfaces Internas de la Lógica de Mediación

La Figura 5, a continuación, muestra el diagrama de las Interfaces Internas de la Lógica de Mediación.

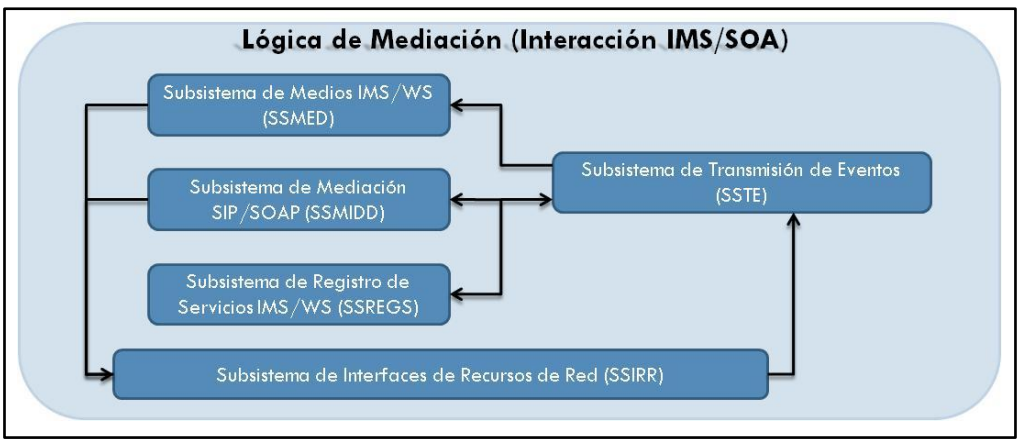


Figura 5. Interfaces Internas de los Subsistemas

- **Subsistema de Interfaces de Recursos de Red – Subsistema de Transmisión de Eventos**
  El *Subsistema de Interfaces de Recursos de Red* recibe los mensajes específicos de los recursos de la red subyacente, los adapta, convirtiéndolos en eventos genéricos, y gestiona sus contextos; finalmente, envía dichos eventos al *Subsistema de Transmisión de Eventos*, a través de una invocación de una operación, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Transmisión de Eventos – Subsistema de Mediación SIP/SOAP**
  El *Subsistema de Transmisión de Eventos* analiza el tipo de evento proveniente del *Subsistema de Interfaces de Recursos de Red* y lo entrega al *Subsistema de Mediación SIP/SOAP* en el caso en que a través de dicho evento se requiera de una mediación, en el plano de señalización y/o control, entre los protocolos de los WS y SIP, a lo cual también se suman los procesos de mediación requeridos para la gestión del registro de servicios IMS/WS. Por esta razón, el *Subsistema de Mediación SIP/SOAP* puede generar nuevos eventos que requieran de la intervención del *Subsistema de Registro de Servicios IMS/WS*, antes de la generación de los resultados finales de mediación. En este caso, dichos eventos son enviados de nuevo al *Subsistema de Transmisión de Eventos* quien, de acuerdo al tipo de evento a transmitir, la jerarquía, y la prioridad de los Subsistemas, se encarga de entregarlos al *Subsistema de Registro de Servicios IMS/WS*. La comunicación entre subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Transmisión de Eventos – Subsistema de Registro de Servicios IMS/WS**
  El *Subsistema de Transmisión de Eventos* entrega al *Subsistema de Registro de Servicios IMS/WS* los eventos provenientes del *Subsistema de Mediación SIP/SOAP* en el caso en que se requiera de la gestión de la información que describe el servicio IMS o el WS, es decir la información de su registro. A su vez, el *Subsistema de Registro de Servicios IMS/WS* puede generar nuevos eventos que requieran de la intervención del *Subsistema de Mediación SIP/WS*, antes de la generación de los resultados finales de mediación. En este caso dichos eventos son enviados de nuevo al *Subsistema de Transmisión de Eventos*, quien se encarga de entregarlos al *Subsistema de Mediación SIP/SOAP*. La comunicación entre estos subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Transmisión de Eventos – Subsistema de Medios IMS/WS**
  El *Subsistema de Transmisión de Eventos* analiza el tipo de evento proveniente del *Subsistema de Interfaces de Recursos de Red* y lo entrega al *Subsistema de Medios IMS/WS* en el caso en que a través de dicho evento se requiera la adaptación de los tipos de medios y formatos de datos Web a los formatos de datos y protocolos de transporte del flujo IP multimedia especificados por la red IMS. La comunicación entre estos subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Medios IMS/WS – Subsistema de Interfaces de Recursos de red**
  Luego de ejecutar la Lógica de Mediación correspondiente, el *Subsistema de Medios IMS/WS* envía los resultados al *Subsistema de Interfaces de Recursos de Red*, el cual recibe

esos eventos genéricos, gestiona sus contextos, y finalmente los adapta en eventos específicos de un recurso de red. La comunicación entre estos dos subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Mediación SIP/SOAP – Subsistema de Interfaces de Recursos de Red**
  Luego de ejecutar la Lógica de Mediación correspondiente, el *Subsistema de Mediación SIP/SOAP* envía los resultados al *Subsistema de Interfaces de Recursos de Red*, el cual recibe esos eventos genéricos, gestiona sus contextos, y finalmente los adapta en eventos específicos de un recurso de red. La comunicación entre estos dos subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

- **Subsistema de Registro de Servicios IMS/WS – Subsistema de Interfaces de Recursos de Red**
  Luego de ejecutar la Lógica de Mediación correspondiente, el *Subsistema de Registro de Servicios IMS/WS* envía los resultados de regreso al *Subsistema de Transmisión de Eventos*, quien luego se encarga de entregarlos al *Subsistema de Mediación SIP/SOAP* para que continúe con el proceso de mediación. La comunicación entre estos subsistemas es llevada a cabo a través de la invocación de operaciones, implementada en cualquier lenguaje que permita la programación tipada de eventos.

3.2.1.2.2. Interfaces Externas de los Subsistemas

A continuación se describen las interfaces que el Subsistema de Interfaces de Recursos de Red posee con el ambiente externo a la Lógica de Mediación, y que se muestran en la Figura 6.

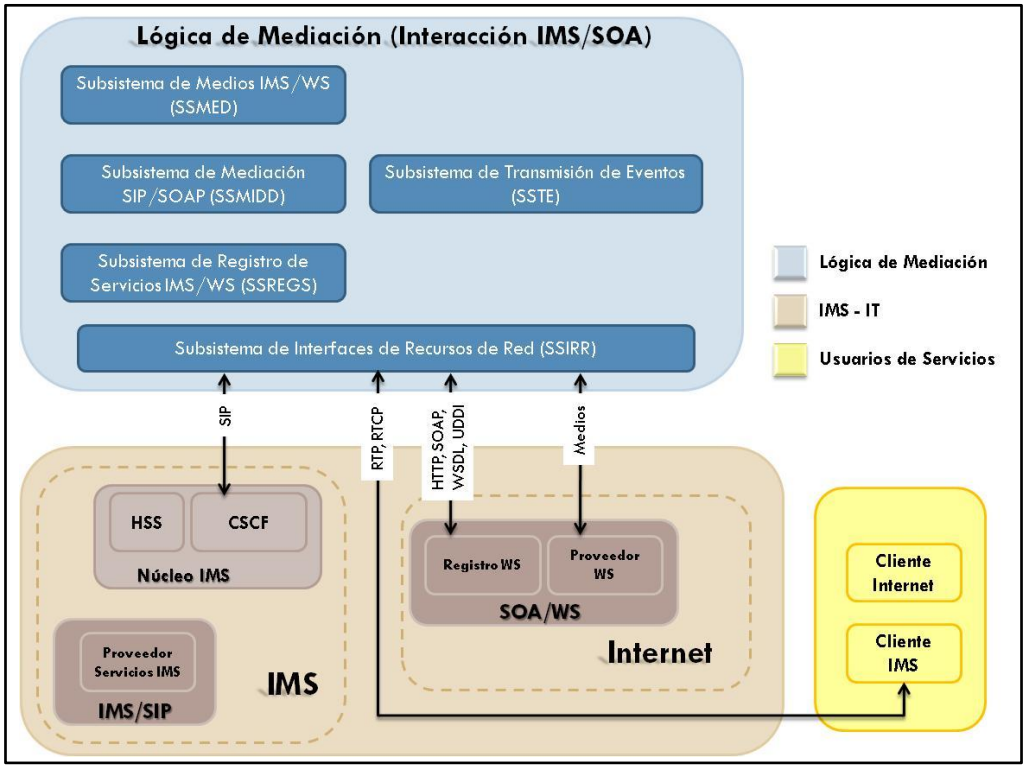


Figura 6. Interfaces externas de los Subsistemas

- **Interfaz SIP**
  Mediante esta interfaz el SSIRR puede dar soporte a los eventos SIP provenientes del Núcleo de IMS, y a su vez puede enviarle a este último los resultados generados al ejecutarse la Lógica de Mediación.

- **Interfaz Medios**
  Mediante esta interfaz el SSIRR puede dar soporte a los eventos de medios provenientes del Cliente IMS, particularmente transportados sobre RTP/RTCP, y a los eventos de medios provenientes del Proveedor de WS. A su vez el SSIRR puede enviar tanto al Cliente IMS como al Proveedor de WS, según sea el caso, los resultados generados al ejecutarse la Lógica de Mediación.

- **Interfaz HTTP, SOAP, WSDL, UDDI**
  Mediante esta interfaz el SSIRR puede dar soporte a los eventos HTTP y SOAP, que transportan otro tipo de protocolos como WSDL y UDDI, los cuales son intercambiados con el componente SOA/WS, al accederse y ejecutarse la Lógica de Mediación.

3.2.1.3. Descripción de los Subsistemas

3.2.1.3.1. SSIRR

Representa la única interfaz de interconexión de la Lógica de Mediación con el medio exterior, es decir, con los recursos de las redes subyacentes (CSCF, SIP, OSA/Parlay, SOAP, bases de datos, etc.) a los que da soporte para su interacción. Proporciona el entorno para la integración de la Lógica de Mediación con la red, y entre sus funciones están:

- Gestionar eventos, entre sus funciones están la suscripción y el arbitraje de eventos.
- Gestionar contextos, entre sus funciones están el inicio, mantenimiento y terminación de los contextos en los cuales se realizan las transacciones de flujos de eventos relacionados.
- Adaptar las interfaces y requerimientos de un recurso de red particular, a las interfaces y requerimientos de la Lógica de Mediación, convirtiendo los mensajes entrantes (protocolos de comunicaciones y eventos de red) en eventos genéricos y con un tipo asociado que los caracterice, los cuales luego son enviados al SSTE para su posterior direccionamiento a los Subsistemas restantes de Mediación, permitiendo así que estos últimos interactúen con los recursos externos respondiendo a los eventos que estos generan.
- Convertir los resultados generados por el SSMIDD, el SSMED, y el SSREGS de eventos genéricos al protocolo o tecnología del recurso de red correspondiente.

Para cumplir con estas funcionalidades el subsistema se subdivide en los componentes mostrados en la Figura 7.

Diagrama de componentes del Subsistema

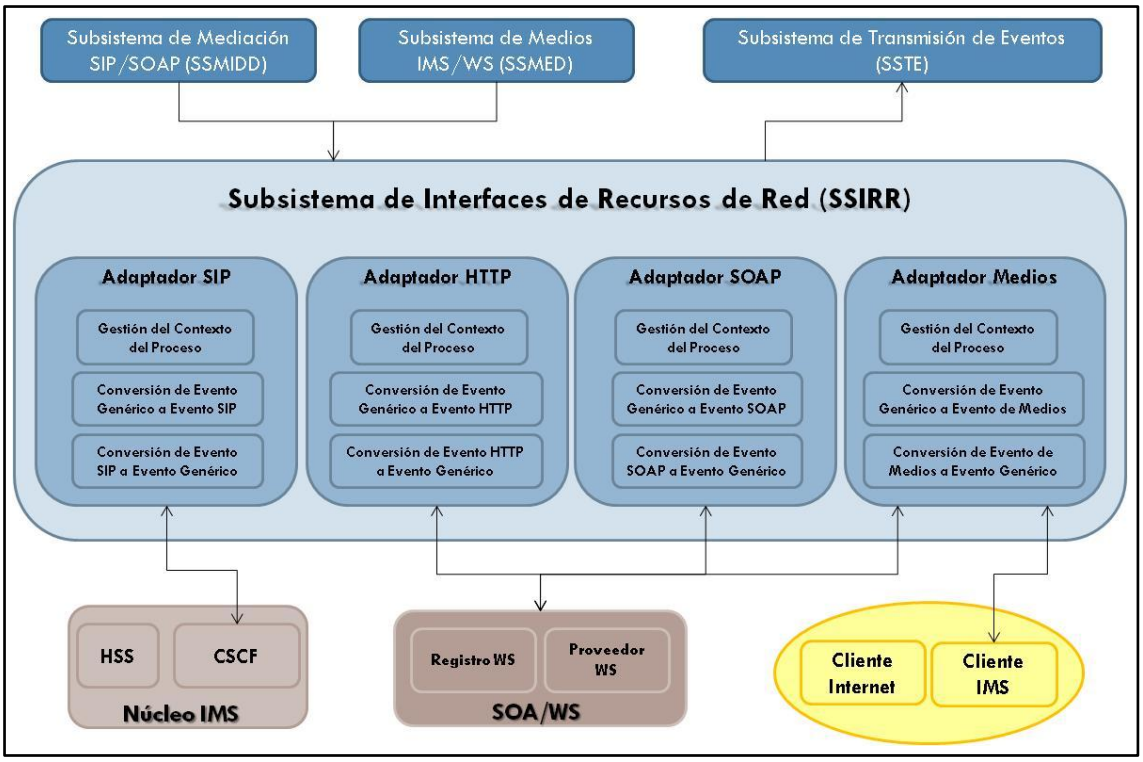


Figura 7. Diagrama de Componentes del Subsistema de Interfaces de Recursos de Red

Componentes del Subsistema

- **Adaptador SIP**: Por un lado, convierte los eventos SIP entrantes, provenientes del Núcleo IMS y específicamente del CSCF, en eventos genéricos con un tipo de evento particular asociado, y luego los entrega al SSTE. Por otro lado, recibe eventos genéricos del SSMIDD, los cuales representan los resultados de la ejecución de los servicios de la Lógica de Mediación, y realiza el proceso de conversión a eventos específicos del protocolo SIP, para finalmente entregarlos al recurso de red correspondiente al CSCF. Adicionalmente a los procesos de conversión de eventos el adaptador realiza acciones para la gestión del contexto de los mismos, tales como: inicio del contexto del proceso donde se van a realizar las transacciones del flujo de eventos relacionados de tipo SIP; análisis del estado de la transacción; análisis del contexto para la determinación de su mantenimiento o terminación, en este último caso, cuando se llega al final de un proceso y no es necesario mantener un medio común tanto para la transmisión de eventos relacionados como para el uso compartido de atributos, el adaptador se encarga de terminar la asociación de los Subsistemas al contexto y luego finalizar el propio contexto.
- **Adaptador HTTP**: Convierte los eventos HTTP entrantes, provenientes del recurso de red SOA/WS, en eventos genéricos con un tipo de evento particular asociado, y luego los entrega al SSTE. Además, recibe eventos genéricos del SSMIDD, los cuales representan los resultados de la ejecución de los servicios de la Lógica de Mediación, y realiza el proceso de conversión a eventos específicos del protocolo HTTP, para finalmente entregarlos al recurso de red correspondiente al componente SOA/WS. Adicionalmente a los procesos de conversión de eventos el adaptador realiza acciones para la gestión del contexto de los

mismos, tales como: inicio del contexto del proceso donde se van a realizar las transacciones del flujo de eventos relacionados de tipo HTTP; análisis del estado de la transacción; análisis del contexto para la determinación de su mantenimiento o terminación, en este último caso, cuando se llega al final de un proceso y no es necesario mantener un medio común tanto para la transmisión de eventos relacionados como para el uso compartido de atributos, el adaptador se encarga de terminar la asociación de los subsistemas al contexto y luego finalizar el contexto mismo.

- **Adaptador SOAP**: Por un lado, convierte los eventos SOAP entrantes, provenientes del recurso de red SOA/WS, en eventos genéricos con un tipo de evento particular asociado, y luego los entrega al SSTE. Por otro lado, recibe eventos genéricos del SSMIDD, los cuales representan los resultados de la ejecución de los servicios de la Lógica de Mediación, y realiza el proceso de conversión a eventos específicos del protocolo SOAP, para finalmente entregarlos al recurso de red correspondiente al componente SOA/WS. Adicionalmente a los procesos de conversión de eventos, el adaptador realiza acciones para la gestión del contexto de los mismos, tales como: inicio del contexto del proceso donde se van a realizar las transacciones del flujo de eventos relacionados de tipo SOAP; análisis del estado de la transacción; análisis del contexto para la determinación de su mantenimiento o terminación, en este último caso, cuando se llega al final de un proceso y no es necesario mantener un medio común tanto para la transmisión de eventos relacionados como para el uso compartido de atributos, el adaptador se encarga de terminar la asociación de los subsistemas al contexto y luego finalizar el contexto mismo.
- **Adaptador de Medios**: Convierte los eventos de recursos de medios entrantes, provenientes del recurso de red SOA/WS, en eventos genéricos con un tipo de evento particular asociado, y luego los entrega al SSTE. Recibe eventos genéricos del SSMED, los cuales representan los resultados de la ejecución de los servicios de la Lógica de Mediación, y realiza el proceso de conversión a eventos específicos de los recursos de medios, para finalmente entregarlos al recurso de red correspondiente al Cliente IMS. Adicionalmente a los procesos de conversión de eventos, el adaptador realiza acciones para la gestión del contexto de los mismos, tales como: inicio del contexto del proceso donde se van a realizar las transacciones del flujo de eventos de medios relacionados; análisis del estado de la transacción; análisis del contexto para la determinación de su mantenimiento o terminación, en este último caso, cuando se llega al final de un proceso y no es necesario mantener un medio común tanto para la transmisión de eventos relacionados como para el uso compartido de atributos, el adaptador se encarga de terminar la asociación de los subsistemas al contexto y luego finalizar el contexto mismo.

Diagrama de interacción de componentes

A continuación se describen los Diagramas de interacción de los componentes del Subsistema de Interfaces de Recursos de Red.

- **Interacción entre el Adaptador SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP (Interacción de IMS con la Lógica de Mediación):**

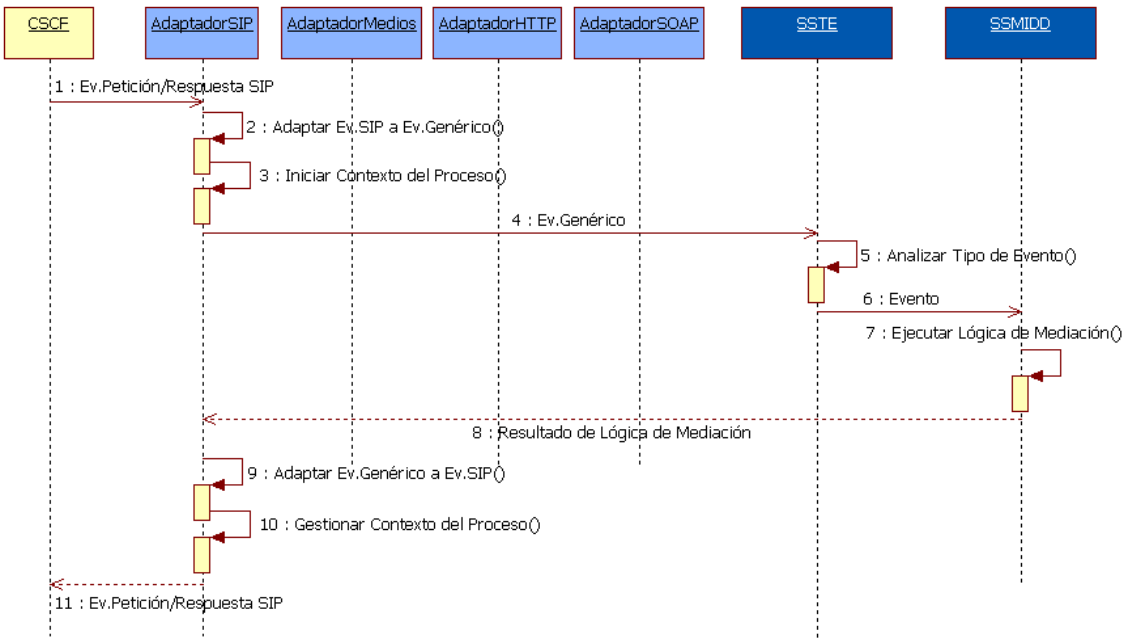


Figura 8. Diagrama de Interacción de Componentes del SSIRR. Interacción entre el Adaptador SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP

Descripción

- El Núcleo IMS, específicamente el CSCF, envía al Subsistema un Evento de Petición/Respuesta SIP. (*Paso 1*)
- El Adaptador SIP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego inicia el contexto de dicho evento, que corresponde a su vez al contexto del proceso cuyo futuro flujo de eventos se relacionan con este evento inicial, y finalmente envía este último al SSTE. (*Pasos 2-4*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué Subsistema está interesado en él. En este caso, como se trata de un evento de Petición/Respuesta del tipo SIP, decide dirigirlo al SSMIDD. (*Pasos 5 y 6*)
- El SSMIDD ejecuta la lógica de mediación necesaria para interpretar la Petición/Respuesta del tipo SIP recibida y generar el mensaje de Petición/Respuesta del tipo SIP correspondiente; finalmente, retorna el resultado al Subsistema. (*Pasos 7 y 8*)
- El Adaptador SIP del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de Petición/Respuesta SIP, y entregado por el Subsistema al componente CSCF de la red IMS. Adicionalmente, el adaptador realiza las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento (análisis del estado, determinación del mantenimiento o terminación del contexto, etc.). (*Pasos 9-11*)

- **Interacción entre el Adaptador de Medios y los Subsistemas de Transmisión de Eventos y de Medios IMS/WS(Adaptación del formato de un Recurso de Medios Web al formato de un Recurso de Medios IMS a través de la Lógica de Mediación):**

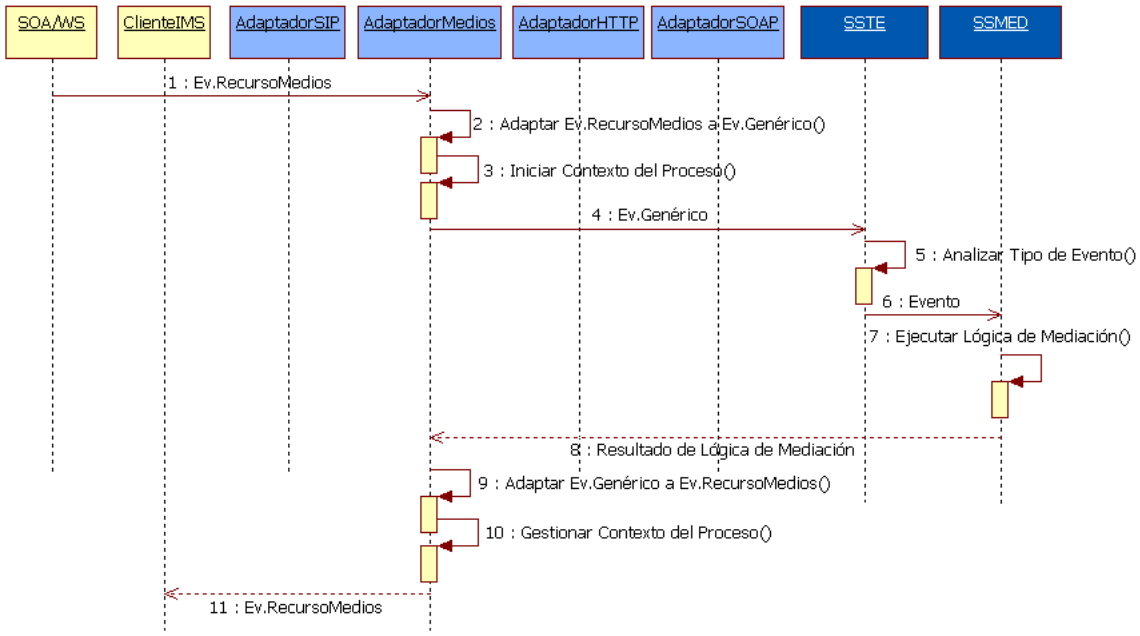


Figura 9. Diagrama de Interacción de Componentes del Subsistema de Interfaces de Recursos de Red. Interacción entre el Adaptador de Medios y los Subsistemas de Transmisión de Eventos y de Medios IMS/WS

Descripción

- El componente SOA/WS envía al Subsistema un Evento de un Recurso de Medios. (*Paso 1*)
- El Adaptador de Medios del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego inicia el contexto de dicho evento, que corresponde a su vez al contexto del proceso cuyo futuro flujo de eventos se relacionan con este evento inicial, y finalmente envía este último al SSTE.(*Pasos 2-4*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué Subsistema está interesado en él. En este caso, como se trata de un evento proveniente de un recurso de medios, decide dirigirlo al SSMED. (*Pasos 5 y 6*)
- El SSMED ejecuta la lógica de mediación necesaria, fija el Cliente IMS que será el destinatario del resultado, y retorna dicho resultado al Subsistema. (*Pasos 7 y 8*)
- El Adaptador de Medios del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de un Recurso de Medios, y entregado por el Subsistema al Cliente IMS fijado. Adicionalmente, el adaptador realiza las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento (análisis del estado, determinación del mantenimiento o terminación del contexto, etc.). (*Pasos 9-11*)

- **Interacción entre los Adaptadores SOAP y SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP (Registro de un Servicio Web en IMS a través de la Lógica de Mediación):**

A continuación se describe el Registro de un WS en IMS a través de la Lógica de Mediación, el cual es el primer proceso que se debe llevar a cabo para la habilitación del uso del WS dentro de la red IMS (Figura 10). Este constituye uno de los principales aportes del presente trabajo de grado de maestría.

Descripción

- El componente SOA/WS, específicamente el Proveedor de WS, envía al Subsistema un Evento de Petición SOAP con los datos necesarios para el proceso de registro del WS en IMS. (*Paso 1*)
- El Adaptador SOAP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego inicia el contexto de dicho evento, que corresponde a su vez al contexto del proceso cuyo futuro flujo de eventos se relacionan con este evento inicial, y finalmente envía este último al SSTE. (*Pasos 2-4*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué Subsistema está interesado en él. En este caso, como se trata de un evento de Petición del tipo SOAP, decide dirigirlo al SSMIDD. (*Pasos 5 y 6*)
- El SSMIDD ejecuta la lógica de mediación necesaria para responder a la petición de Registro del WS en IMS, y retorna el resultado al Subsistema.(*Pasos 7 y 8*)
- El Adaptador SIP del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de Petición SIP del tipo REGISTER, y entregado por el Subsistema al componente CSCF de la red IMS. Adicionalmente, el adaptador realiza las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento (análisis del estado, determinación del mantenimiento o terminación del contexto, etc.). (*Pasos 9-11*)
- El Núcleo IMS, específicamente el CSCF, ejecuta el proceso de registro del WS en IMS para que su uso esté disponible en los procesos de creación de los servicios convergentes IMS/SOA, y envía al Subsistema un Evento de Respuesta SIP del tipo 200 OK. (*Pasos 12 y 13*)
- El Adaptador SIP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego ejecuta las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento, y finalmente envía este último al SSTE. (*Pasos 14-16*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué Subsistema está interesado en él. En este caso, como se trata de un evento de Respuesta del tipo SIP, decide dirigirlo al SSMIDD. (*Pasos 17 y 18*)
- El SSMIDD ejecuta la lógica de mediación necesaria para interpretar el resultado del registro y generar el mensaje de comprobación de registro exitoso del WS en IMS; finalmente, retorna este resultado al Subsistema. (*Pasos 19 y 20*)
- El Adaptador SOAP del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de Respuesta SOAP exitosa, y entregado por el Subsistema al Proveedor de WS del componente SOA/WS. Adicionalmente, el adaptador realiza las acciones correspondientes

para la gestión del contexto del proceso que involucra dicho evento; en este caso, como el proceso de Registro del WS en IMS ha concluido entonces el adaptador termina tanto la asociación de los subsistemas con el contexto como el contexto mismo. (*Pasos 21-23*)

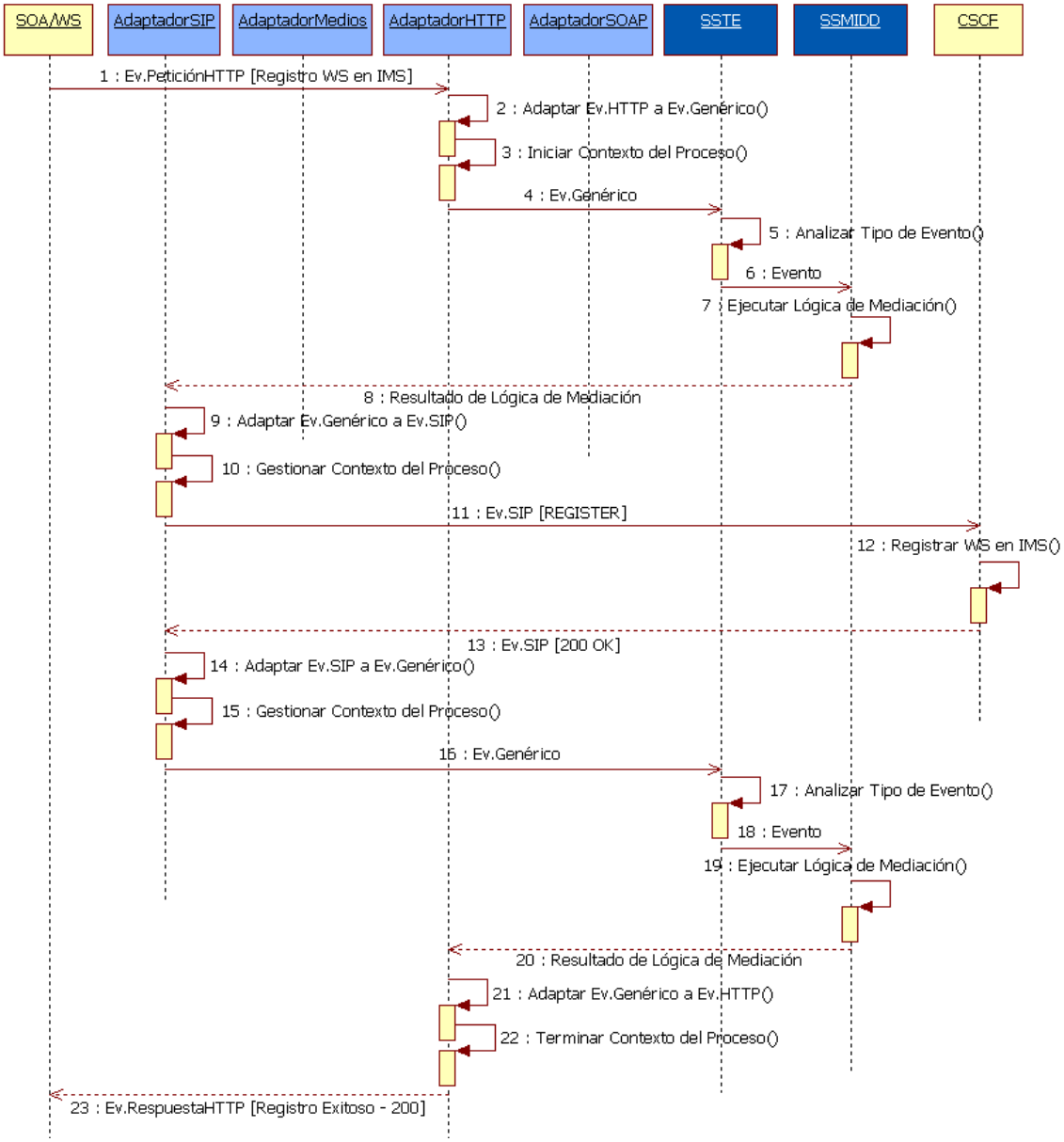


Figura 10. Diagrama de Interacción de Componentes del Subsistema de Interfaces de Recursos de Red. Interacción entre los Adaptadores HTTP y SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP

- **Interacción entre los Adaptadores SOAP y SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP (Inicio de sesión entre un Servicio de la red IMS y un Servicio Web, a través de la Lógica de Mediación, para la creación de un Servicio Convergente IMS/SOA):**

A continuación se describe el Inicio de sesión entre un Servicio de la red IMS y un WS, a través de la Lógica de Mediación, el cual es un proceso que se debe llevar a cabo para la creación de un Servicio Convergente IMS/SOA (Figura 11). Este constituye uno de los principales aportes del presente trabajo de grado de maestría.

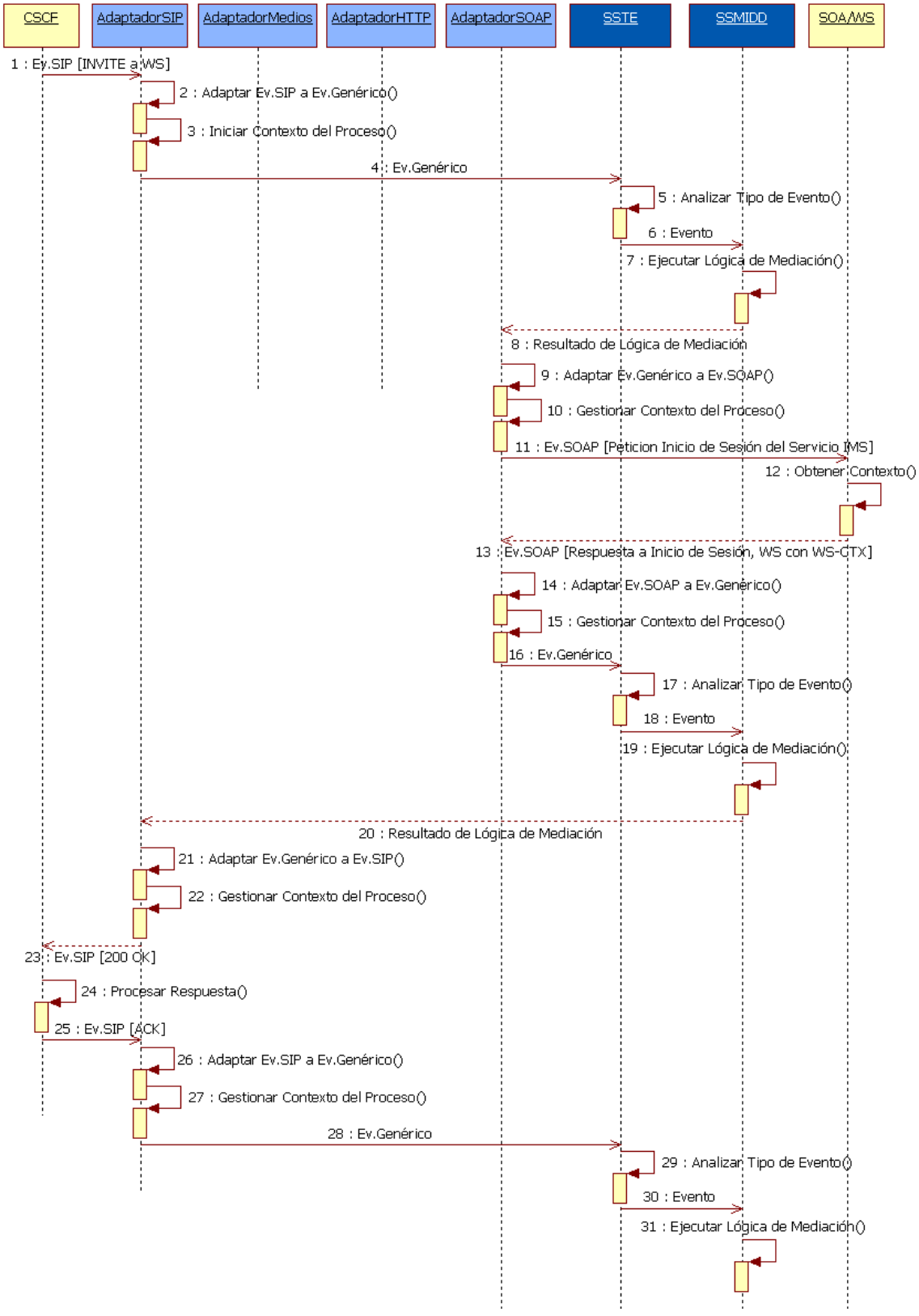


Figura 11. Diagrama de Interacción de Componentes del Subsistema de Interfaces de Recursos de Red. Interacción entre los Adaptadores SOAP y SIP y los Subsistemas de Transmisión de Eventos y de Mediación SIP/SOAP

Descripción

- El Núcleo IMS, específicamente el CSCF, envía al Subsistema un Evento de Petición SIP del tipo INVITE con los datos necesarios para el proceso de inicio de sesión de un Servicio IMS con un WS previamente registrado en IMS. (*Paso 1*)
- El Adaptador SIP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego inicia el contexto de dicho evento, que corresponde a su vez al contexto del proceso cuyo futuro flujo de eventos se relacionan con este evento inicial, y finalmente envía este último al SSTE. (*Pasos 2-4*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué subsistema está interesado en él. En este caso, como se trata de un evento de Petición del tipo SIP, decide dirigirlo al SSMIDD. (*Pasos 5 y 6*)
- El SSMIDD ejecuta la lógica de mediación necesaria para interpretar la petición de Inicio de Sesión del Servicio IMS con el WS y generar el mensaje de petición de inicio de sesión hacia el WS requerido; finalmente, retorna este resultado al Subsistema. (*Pasos 7 y 8*)
- El Adaptador SOAP del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de Petición SOAP para el inicio de una sesión, y entregado por el Subsistema al componente SOA/WS. Adicionalmente, el adaptador realiza las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento (análisis del estado, determinación del mantenimiento o terminación del contexto, etc.). (*Pasos 9-11*)
- El componente SOA/WS ejecuta el proceso de inicio de sesión del Servicio IMS con el WS especificado. Durante este proceso, primero el WS comprueba, por medio del Gestor de Contexto, si en la petición existe un Contexto (en el encabezado del mensaje SOAP), en este caso, como no existe el contexto en el dominio del WS, el Gestor de Contexto retorna al WS un valor de NULL y por consiguiente necesita crearse uno nuevo. La Creación del nuevo Contexto se inicia al enviar una petición *begin* desde el WS al Servicio de Contexto, a lo que este último crea el Contexto y propaga la petición hacia la lista de los Servicios de Ciclo de Vida de la Actividad (Activity Lifecycle Service, ALS) registrados, los cuales necesitan ser notificados para las acciones *begin* y *complete*. Dado que sólo está registrado un ALS éste es invocado y extiende los datos del Contexto del WS con la información del inicio de sesión del Servicio IMS. Como último paso en la creación del Contexto del WS el ALS retorna el Contexto extendido al Servicio de Contexto, y a su vez el Servicio de Contexto retorna el Contexto, basado en la especificación WS-CTX, al WS que lo solicitó, terminando así el proceso de inicio de sesión entre el Servicio IMS y el WS. Finalmente, el componente SOA/WS envía al Subsistema un Evento SOAP de Respuesta al inicio exitoso de la sesión, junto con el Contexto del tipo WS-CTX creado. (*Pasos 12 y 13*)
- El Adaptador SOAP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego ejecuta las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento, y finalmente envía este último al SSTE. (*Pasos 14-16*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué Subsistema está interesado en él. En este caso, como se trata de un evento de Respuesta del tipo SOAP, decide dirigirlo al SSMIDD. (*Pasos 17 y 18*)
- El SSMIDD ejecuta la lógica de mediación necesaria para interpretar el resultado de inicio exitoso de sesión junto con el Contexto del tipo WS-CTX, y generar como resultado el

mensaje de respuesta exitosa hacia IMS; finalmente, retorna este resultado al Subsistema. (*Pasos 19 y 20*)

- El Adaptador SIP del Subsistema recibe el resultado del proceso de mediación. Este resultado, representado por un evento genérico, es ahora adaptado a un evento específico de Respuesta SIP del tipo 200 OK, y entregado por el Subsistema al CSCF del Núcleo de IMS. Adicionalmente, el adaptador realiza las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento (análisis del estado, determinación del mantenimiento o terminación del contexto, etc.). (*Pasos 21-23*)
- El Núcleo IMS, específicamente el CSCF, procesa la respuesta SIP recibida y envía al Subsistema un Evento de Petición SIP del tipo ACK para la confirmación de que el Servicio IMS recibió la respuesta final al mensaje del tipo INVITE, es decir la confirmación de una sesión iniciada exitosamente. (*Pasos 24 y 25*)
- El Adaptador SIP del Subsistema recibe éste evento y realiza su adaptación a un evento genérico. Luego ejecuta las acciones correspondientes para la gestión del contexto del proceso que involucra dicho evento, y finalmente envía este último al SSTE. (*Pasos 26-28*)
- El SSTE analiza el tipo de evento y de acuerdo a éste examina qué subsistema está interesado en él. En este caso, como se trata de un evento de Petición del tipo SIP, decide dirigirlo al SSMIDD. (*Pasos 29 y 30*)
- El SSMIDD ejecuta la lógica de mediación necesaria para interpretar el evento de confirmación de inicio exitoso de sesión. Finalmente, como el proceso de Inicio de una Sesión ha concluido, cada adaptador involucrado realiza las acciones correspondientes para la gestión de los contextos respectivos, terminando tanto la asociación de los subsistemas con ellos, como terminando los contextos en sí. (*Paso 31*)

3.2.1.3.2. SSTE

Es el núcleo del direccionamiento de eventos de la Lógica de Mediación. Está encargado básicamente, de que cada uno de los Subsistemas de Mediación se comuniquen entre sí y con la infraestructura de recursos subyacente. Sus funciones son:

- Analizar el tipo de evento enviado por el SSIRR, y verificar el contexto del proceso en el cual se realiza la transacción del evento. De acuerdo al tipo de evento examinar qué Subsistema de Mediación restante está interesado en él, establecerlo y asociarlo a su contexto, si aún no lo está, para que el Subsistema pueda tanto recibir como enviar eventos relacionados con el mismo proceso, y también pueda compartir sus atributos con los demás subsistemas asociados al contexto.
- Analizar el tipo de evento enviado ya sea por el SSMIDD o por el SSREGS, y verificar el contexto del proceso en el cual se realiza la transacción del evento. De acuerdo al tipo de evento examinar qué subsistema de Mediación restante está interesado en él, establecerlo y asociarlo a su contexto, si aún no lo está, para que el Subsistema pueda tanto recibir como enviar eventos relacionados con el mismo proceso, y también pueda compartir sus atributos con los demás subsistemas asociados al contexto.
- Realizar el direccionamiento y la entrega de eventos teniendo en cuenta la jerarquía y prioridad de los Subsistemas.

- Asegurar la correcta transmisión de cada evento desde su productor hasta su consumidor, desacoplando estos dos últimos y dejando el control de los Subsistemas a la Lógica de Mediación. Esto lo consigue asociando al contexto del proceso, en el cual se realiza la transacción del evento, el subsistema interesado en dicho tipo de evento, y separándolo cuando se llegue al final del proceso.

Para cumplir con estas funcionalidades el subsistema se subdivide en los componentes mostrados en la Figura 12.

Diagrama de componentes del Subsistema

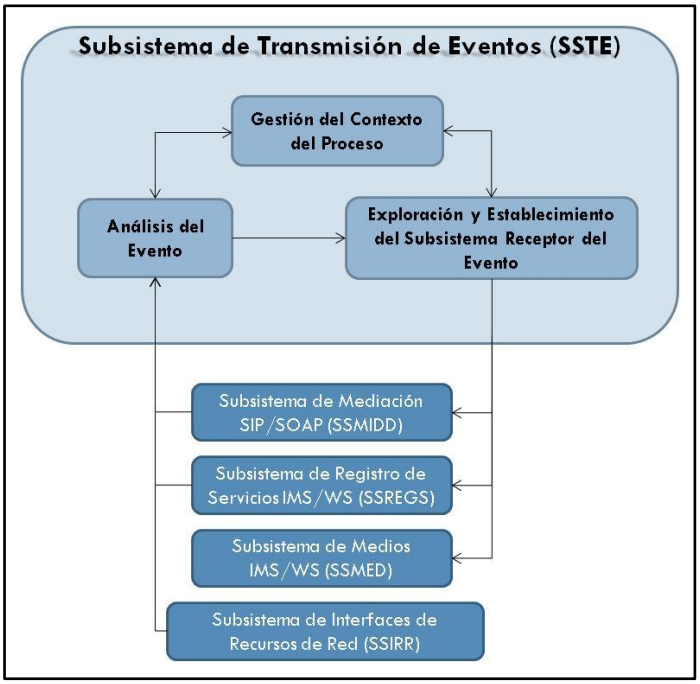


Figura 12. Diagrama de Componentes del Subsistema de Transmisión de Eventos

Componentes del Subsistema

- **Análisis del Evento**: Realiza el análisis del tipo de evento recibido, que puede ser SIP, HTTP, SOAP, de medios, y genérico, y el cual puede provenir tanto del SSIRR como de el SSMIDD y del SSREGS. Luego verifica el contexto del proceso, en el cual se realiza la transacción del evento, usando las funcionalidades del componente de Gestión del Contexto del Proceso. Finalmente, envía el evento junto con la información de su tipo y su contexto al componente de Exploración y Establecimiento del Subsistema Receptor del Evento.
- **Gestión del Contexto del Proceso**: Este componente brinda a los otros dos componentes las funcionalidades relacionadas con la gestión del contexto de los eventos. Sus funciones principales son: la verificación del contexto de un evento, en la cual se comprueba si el evento que se está transmitiendo pertenece a un proceso que tiene establecido su correspondiente contexto; el análisis y establecimiento de la asociación de un Subsistema al contexto de un evento, en la cual se analiza si un Subsistema dado posee o no una relación de asociación con el contexto de proceso determinado, y en el caso de no tenerla se establece, con el fin de que el Subsistema pueda compartir sus atributos, además de recibir

y enviar eventos dentro del mismo proceso, es decir, de comunicar eventos relacionados por medio del uso de un contexto común.

- **Exploración y Establecimiento del Subsistema Receptor del Evento:** Realiza una exploración de los Subsistemas que han registrado un interés en el tipo de evento que se está transmitiendo, y selecciona el Receptor de acuerdo a la jerarquía y orden de prioridad establecida entre los Subsistemas. En esta arquitectura el SSMIDD puede recibir eventos de tipo SIP, HTTP y SOAP; el SSREGS sólo puede recibir eventos de petición de registro de servicios convergentes IMS/WS provenientes del SSMIDD; y el SSMED sólo puede recibir eventos de medios. Luego de seleccionar el Subsistema Receptor del evento, y haciendo uso del componente de Gestión del Contexto del Proceso, verifica si el subsistema está asociado al contexto del evento, sino lo está realiza dicha asociación, y finalmente transmite el evento al Subsistema Receptor.

Interfaces Internas

- **Análisis del Evento – Gestión del Contexto del Proceso**: Por medio de esta interfaz el componente de Gestión del Contexto del Proceso proporciona al componente de Análisis del Evento la función de verificación del contexto del proceso en el cual se realiza la transacción del evento.
- **Análisis del Evento – Exploración y Establecimiento del Subsistema Receptor del Evento**: Interfaz que comunica estos dos componentes, permitiendo el envío de los resultados del proceso de análisis del tipo de evento y verificación del contexto del evento desde el componente de Análisis del Evento hacia el componente de Exploración y Establecimiento del Subsistema Receptor del Evento.
- **Exploración y Establecimiento del Subsistema Receptor del Evento – Gestión del Contexto del Proceso**: Por medio de esta interfaz el componente de Gestión del Contexto del Proceso proporciona al componente de Exploración y Establecimiento del Subsistema Receptor del Evento las funciones de verificación de la asociación entre el subsistema y el contexto dados, y de establecimiento de la relación de asociación entre el subsistema y el contexto en el caso de que no exista.

Diagrama de interacción de componentes

A continuación se describe el Diagrama de interacción de componentes del Subsistema de Transmisión de Eventos (Figura 13).

Descripción

- El componente SOA/WS envía al SSIRR un Evento de Petición SOAP con la información de registro de un Servicio (IMS o Web). (*Paso 1*)
- El SSIRR recibe el evento, realiza su adaptación a un evento genérico, inicia el contexto del proceso, y luego envía el Evento al Subsistema. (*Pasos 2 y 3*)
- El componente de Análisis del Evento del Subsistema recibe el Evento Genérico y empieza por analizar su tipo, que en este caso es SOAP. Luego envía este evento al componente de Gestión del Contexto del Evento. (*Pasos 4 y 5*)
- El componente de Gestión del Contexto del Proceso del Subsistema verifica entonces el Contexto del proceso en el cual se realiza la transacción del evento, y lo retorna al

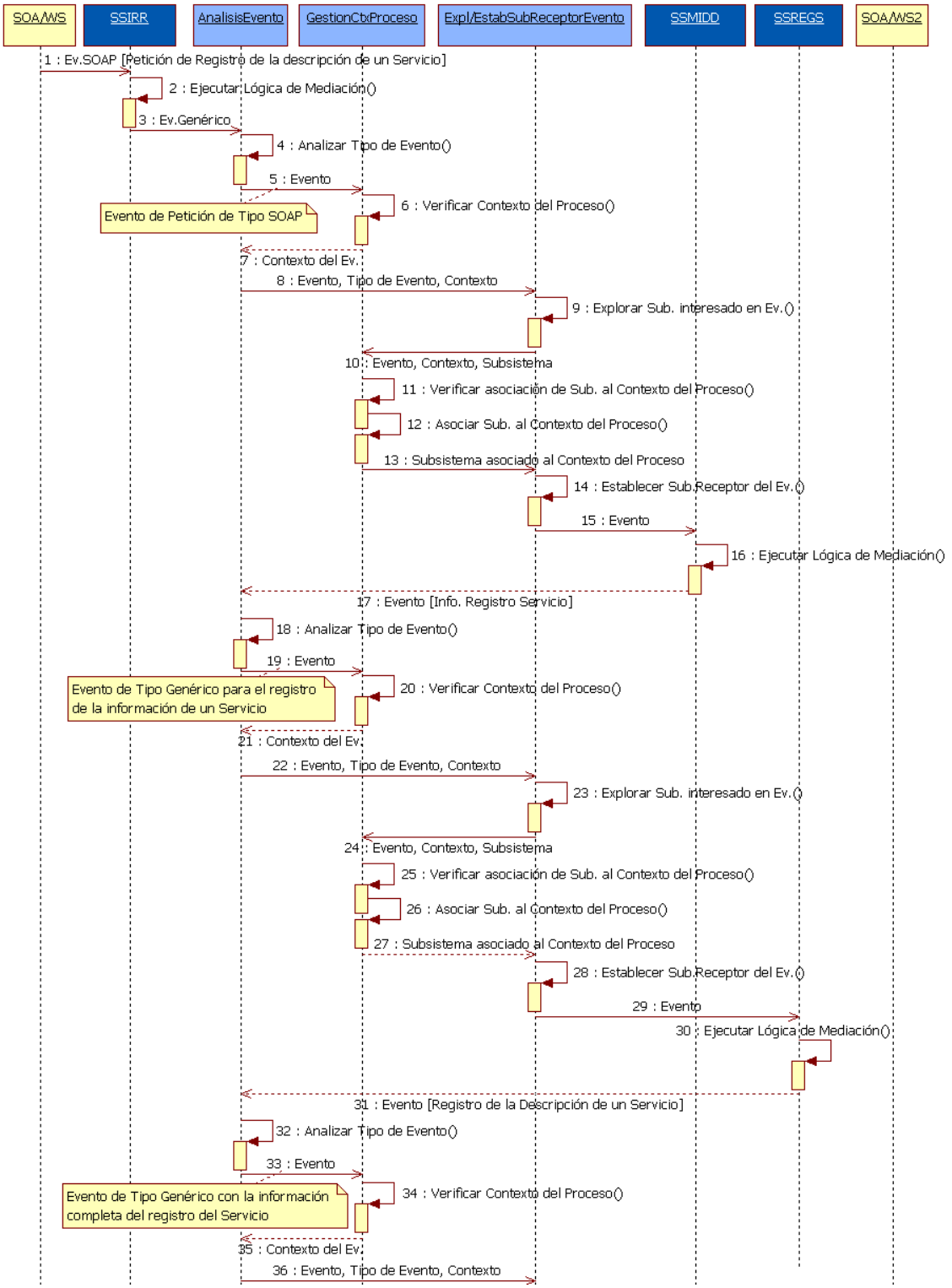
componente de Análisis del Evento para que continúe el proceso de transmisión.(*Pasos 6 y 7*)

- El componente de Análisis del Evento, una vez recibe el Contexto del Evento, lo reenvía junto con el Evento mismo, y su Tipo, al componente de Exploración y Establecimiento del Subsistema Receptor del Evento. (*Paso 8*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe estos elementos y en base a ellos realiza una exploración de los Subsistemas que han registrado un interés en el tipo de evento que se está transmitiendo. De acuerdo a la jerarquía y orden de prioridad establecida entre los Subsistemas el resultado de la exploración es la selección del SSMIDD como Subsistema Receptor. A continuación el componente envía el Evento, su Contexto y el Subsistema interesado, al componente de Gestión del Contexto del Proceso para verificar, establecer, o mantener, la relación entre estos tres elementos. (*Pasos 9 y 10*)
- El componente de Gestión del Contexto del Proceso del Subsistema inicialmente verifica si el Subsistema interesado ya está asociado al Contexto del Proceso en el cual se realiza la transacción del evento. Como en este caso no existe tal relación entonces el componente asocia el Subsistema Receptor al Contexto del Proceso, y retorna al componente de Exploración y Establecimiento del Subsistema Receptor del Evento el Subsistema Receptor asociado.(*Pasos 11-13*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe el Subsistema Receptor, asociado correctamente al Contexto del Proceso, y es en este momento que establece al SSMIDD como Subsistema Receptor del Evento. Finalmente este componente del Subsistema envía el Evento al destino fijado. (*Pasos 14 y 15*)
- El SSMIDD recibe el Evento y ejecuta la lógica de mediación necesaria para interpretar la Petición del tipo SOAP y obtener la información de registro del Servicio. A continuación, crea un nuevo proceso y dentro de él genera un nuevo Evento de Petición con los datos del registro, inicia su contexto, y finalmente envía dicho Evento al SSTE para que sea dirigido al SSREGS, el cual se encargará de su procesamiento. (*Pasos 16 y 17*)
- El componente de Análisis del Evento del Subsistema recibe el Evento y analiza su tipo, que en este caso se trata de un evento genérico para el registro de un Servicio. Luego envía este evento al componente de Gestión del Contexto del Proceso. (*Pasos 18 y 19*)
- El componente de Gestión del Contexto del Proceso del Subsistema verifica entonces el Contexto del proceso en el cual se realiza la transacción del evento, y lo retorna al componente de Análisis del Evento para que continúe el proceso de transmisión.(*Pasos 20 y 21*)
- El componente de Análisis del Evento, una vez recibe el Contexto del Evento, lo reenvía junto con el Evento mismo, y su Tipo, al componente de Exploración y Establecimiento del Subsistema Receptor del Evento. (*Paso 22*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe estos elementos y en base a ellos realiza una exploración de los Subsistemas que han registrado un interés en el tipo de evento que se está transmitiendo. De acuerdo a la jerarquía y orden de prioridad establecida entre los Subsistemas el resultado de la exploración es la selección del SSREGS como Subsistema Receptor. A continuación el componente envía el Evento, su Contexto y el Subsistema interesado, al componente de

Gestión del Contexto del Proceso para verificar, establecer, o mantener, la relación entre estos tres elementos. (*Pasos 23 y 24*)

- El componente de Gestión del Contexto del Proceso del Subsistema inicialmente verifica si el subsistema interesado ya está asociado al Contexto del Proceso en el cual se realiza la transacción del evento. Como en este caso no existe tal relación entonces el componente asocia el Subsistema Receptor con el Contexto del Proceso, y retorna al componente de Exploración y Establecimiento del Subsistema Receptor del Evento el Subsistema Receptor asociado. (*Pasos 25-27*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe el Subsistema Receptor, asociado correctamente al Contexto del Proceso, y es en este momento que establece al SSREGS como Subsistema Receptor del Evento. Finalmente este componente del Subsistema envía el Evento al destino fijado. (*Pasos 28 y 29*)
- El SSREGS recibe el Evento y ejecuta la lógica de mediación necesaria para responder a la Petición de generación del registro del Servicio, basado en el modelo de registro de los Servicios Web. El Resultado de la mediación es enviado como Evento al SSTE para que luego sea transmitido como respuesta al SSMIDD. (*Pasos 30 y 31*)
- El componente de Análisis del Evento del Subsistema recibe el Evento y analiza su tipo, que en este caso se trata de un evento genérico para el registro de la información completa de descripción de un Servicio. Luego envía este evento al componente de Gestión del Contexto del Proceso. (*Pasos 32 y 33*)
- El componente de Gestión del Contexto del Proceso del Subsistema verifica entonces el Contexto del proceso en el cual se realiza la transacción del evento, y lo retorna al componente de Análisis del Evento para que continúe el proceso de transmisión.(*Pasos 34 y 35*)
- El componente de Análisis del Evento, una vez recibe el Contexto del Evento, lo reenvía junto con el Evento mismo, y su Tipo, al componente de Exploración y Establecimiento del Subsistema Receptor del Evento. (*Paso 36*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe estos elementos y en base a ellos realiza una exploración de los Subsistemas que han registrado un interés en el tipo de evento que se está transmitiendo. De acuerdo a la jerarquía entre los Subsistemas el resultado de la exploración es la selección del SSMIDD como Subsistema Receptor. A continuación, el componente envía el Evento, su Contexto y el Subsistema interesado, al componente de Gestión del Contexto del Proceso para verificar, establecer, o mantener, la relación entre estos tres elementos. (*Pasos 37 y 38*)
- El componente de Gestión del Contexto del Proceso del Subsistema inicialmente verifica si el Subsistema interesado ya está asociado al Contexto del Proceso en el cual se realiza la transacción del evento. Como en este caso si existe tal relación entonces el componente mantiene la asociación del Subsistema Receptor con el Contexto del Proceso, y retorna al componente de Exploración y Establecimiento del Subsistema Receptor del Evento el Subsistema Receptor asociado. (*Pasos 39-41*)
- El componente de Exploración y Establecimiento del Subsistema Receptor del Evento recibe el Subsistema Receptor asociado y es en este momento que establece al SSMIDD como Subsistema Receptor del Evento. Finalmente este componente del Subsistema envía el Evento al destino fijado. (*Pasos 42 y 43*)

- El SSMIDD recibe el Evento y ejecuta la lógica de mediación necesaria para generar el Evento de Respuesta del tipo SOAP y así confirmar la Publicación exitosa de la descripción del Servicio (IMS o Web). Finalmente envía dicho Evento al SSIRR. (*Pasos 44 y 45*)
- El SSIRR recibe el evento, realiza su adaptación a un evento de tipo SOAP, lo envía al componente SOA/WS, luego termina la asociación de los subsistemas con el Contexto del Proceso y finalmente termina el contexto mismo. (*Pasos 46 y 47*)



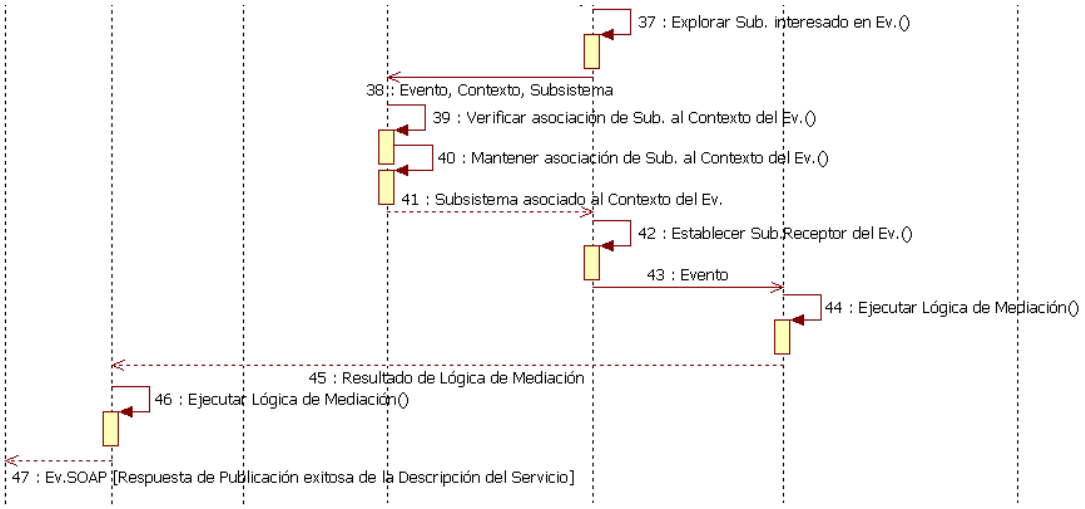


Figura 13. Diagrama de Interacción de Componentes del Subsistema de Transmisión de Eventos. Interacción entre los componentes de Análisis de Evento, de Gestión del Contexto del Evento, y de Exploración y Establecimiento del Subsistema Receptor del Evento y con los Subsistemas de Interfaces de Recursos de Red, de Mediación SIP/SOAP, y de Registro de Servicios IMS/WS

3.2.1.3.3. SSMIDD

Es el subsistema encargado de la mediación a nivel de señalización y de control entre los servicios basados en IMS y los WS. Este subsistema cumple con las funciones:

- Enviar, recibir, y procesar Eventos, por una parte, de Tipo SIP y SOAP de la infraestructura de red subyacente, y, por otra parte, de tipo Genérico desde y hacia el SSREGS.
- Procesar cada evento recibido, tanto de los procesos de señalización como de control, dependiendo de su tipo.
- Adaptar eventos de tipo SIP a solicitudes de tipo SOAP y genéricas.
- Adaptar eventos de tipo SOAP a solicitudes de tipo SIP y genéricas.
- Responder a eventos de tipo SIP y SOAP con eventos directos tipo SIP y SOAP, respectivamente.
- Generar solicitudes de tipo genérico.
- Transmitir los resultados obtenidos de los procesos de análisis y mediación hacia el SSIRR, para que, dependiendo de su tipo, sean transmitidos al elemento de la red subyacente correspondiente.

Para cumplir con estas funcionalidades el subsistema se subdivide en los componentes mostrados en la Figura 14.

Diagrama de componentes del Subsistema

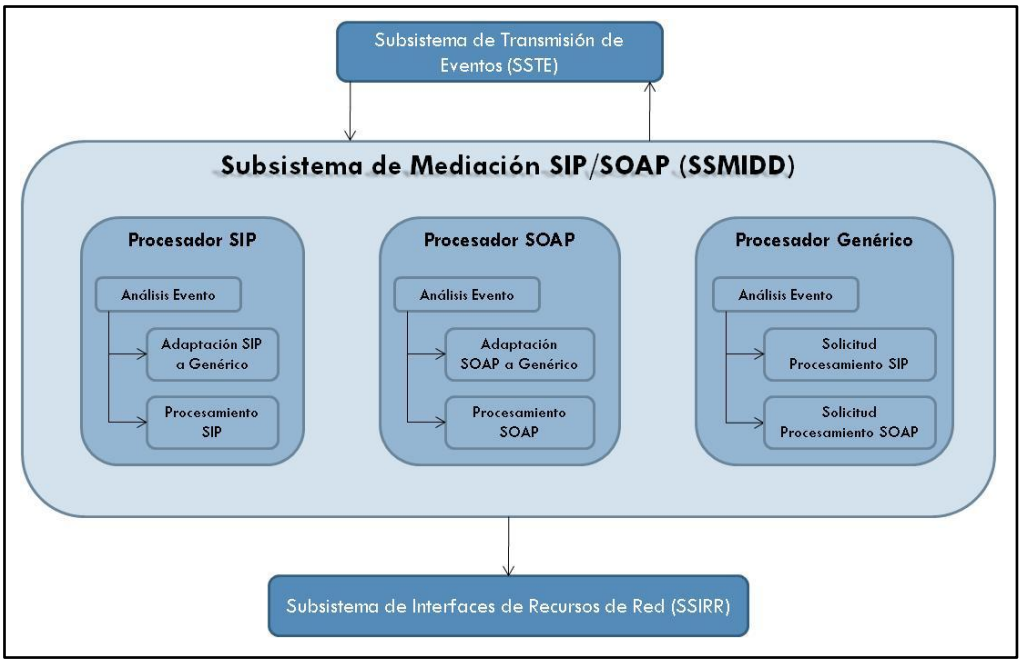


Figura 14. Diagrama de Componentes del Subsistema de Mediación SIP/SOAP

Componentes del Subsistema

- **Procesador SIP**: El SSMIDD recibe eventos de tipo SIP, provenientes del SSTE, y entrega dichos eventos al Procesador SIP para que se encargue del análisis, adaptación, y generación de respuestas a este tipo de eventos. La primera tarea de este componente es el análisis del evento recibido, la cual consiste, por una parte, en la exploración del evento para la determinación del tipo de mensaje SIP (petición o respuesta) junto con su valor y contenido, y, por otra parte, en el análisis del contexto del proceso donde se está realizando la transacción del evento y en el almacenamiento y/o recuperación de los datos persistentes relacionados con el proceso particular. El resultado de este análisis es la determinación de la tarea que se debe realizar a continuación, la cual puede ser la adaptación del evento de tipo SIP a un evento genérico, o el procesamiento de eventos específicos SIP para la generación de una respuesta de este tipo de protocolo de comunicaciones hacia la red subyacente. La adaptación del evento tipo SIP a un evento de tipo genérico se llevará a cabo cuando se requiera de la mediación en la comunicación, ya sea de señalización o control, desde la red IMS hacia el dominio SOA para: el inicio de sesión entre un servicio IMS y un WS, para la creación de un Servicio Convergente IMS/SOA; el acceso a una funcionalidad de un WS, publicado en UDDI, desde un servicio IMS. El procesamiento de eventos SIP, se llevará a cabo para la generación de una respuesta hacia la red subyacente, específica a este protocolo de comunicación, la cual puede tener su origen de solicitud tanto en el Procesador Genérico como en la misma red subyacente.
- **Procesador SOAP**: El SSMIDD recibe eventos de tipo SOAP, provenientes del SSTE, y los entrega al Procesador SOAP para que se encargue del análisis, adaptación, y generación de las respuestas respectivas. La primera tarea de este componente es el análisis del evento recibido, la cual consiste, por una parte, en la exploración del evento para la determinación

del tipo de mensaje SOAP (solicitud o respuesta) y la información tanto de infraestructura y de valor añadido, presente en el contenido de la cabecera de los mensajes SOAP, como de la aplicación, presente en el cuerpo del mensaje. Por otra parte, en esta tarea se realiza el análisis del contexto del proceso donde se está realizando la transacción del evento y también el almacenamiento y/o recuperación de los datos persistentes relacionados con el proceso particular. El resultado de este análisis es la determinación de la tarea que se debe realizar a continuación, la cual puede ser la adaptación del evento de tipo SOAP a un evento genérico, o el procesamiento de eventos específicos SOAP para la generación de una respuesta de este tipo de protocolo de comunicaciones hacia la red subyacente. La adaptación del evento tipo SOAP a un evento de tipo genérico se llevará a cabo cuando se requiera de la mediación en la comunicación, ya sea de señalización o control, desde el dominio SOA hacia la red IMS para: el inicio de sesión entre un servicio IMS y un WS para la creación de un Servicio Convergente IMS/SOA; el registro de un servicio IMS o Web en UDDI; el acceso a una funcionalidad de un WS, publicado en UDDI, desde un Servicio IMS. El procesamiento de eventos SOAP, se llevará a cabo para la generación de una respuesta hacia la red subyacente, específica a este protocolo de comunicación, la cual puede tener su origen de solicitud tanto en el Procesador Genérico como en la misma red subyacente.

- **Procesador Genérico**: El SSMIDD recibe eventos de tipo genérico, provenientes del SSTE, y los entrega al Procesador Genérico para que se encargue del análisis, adaptación, y generación de solicitudes orientadas a cualquiera de los Procesadores restantes según sea el caso. La primera tarea de este componente es el análisis del evento recibido, la cual consiste, por una parte, en la exploración del evento para la determinación de la información que contiene, y por otra, en el análisis del contexto del proceso donde se está realizando la transacción del evento y también el almacenamiento y/o recuperación de los datos persistentes relacionados con el proceso particular. El resultado de este análisis es la determinación de la tarea que se debe realizar a continuación, la cual puede ser la generación de una solicitud de procesamiento SIP o SOAP hacia el Procesador del tipo de protocolo específico, según sea el requerimiento del evento de recibido.

El SSMIDD no posee interfaces internas entre sus Procesadores, ya que todo evento intercambiado dentro de la Lógica de Mediación es enviado hacia y desde el SSTE.

Diagrama de interacción de componentes

A continuación se describe el Diagrama de interacción de componentes del Subsistema de Mediación SIP/SOAP (Figura 15), para el Registro de un Servicio Web en IMS a través de la Lógica de Mediación.

Descripción

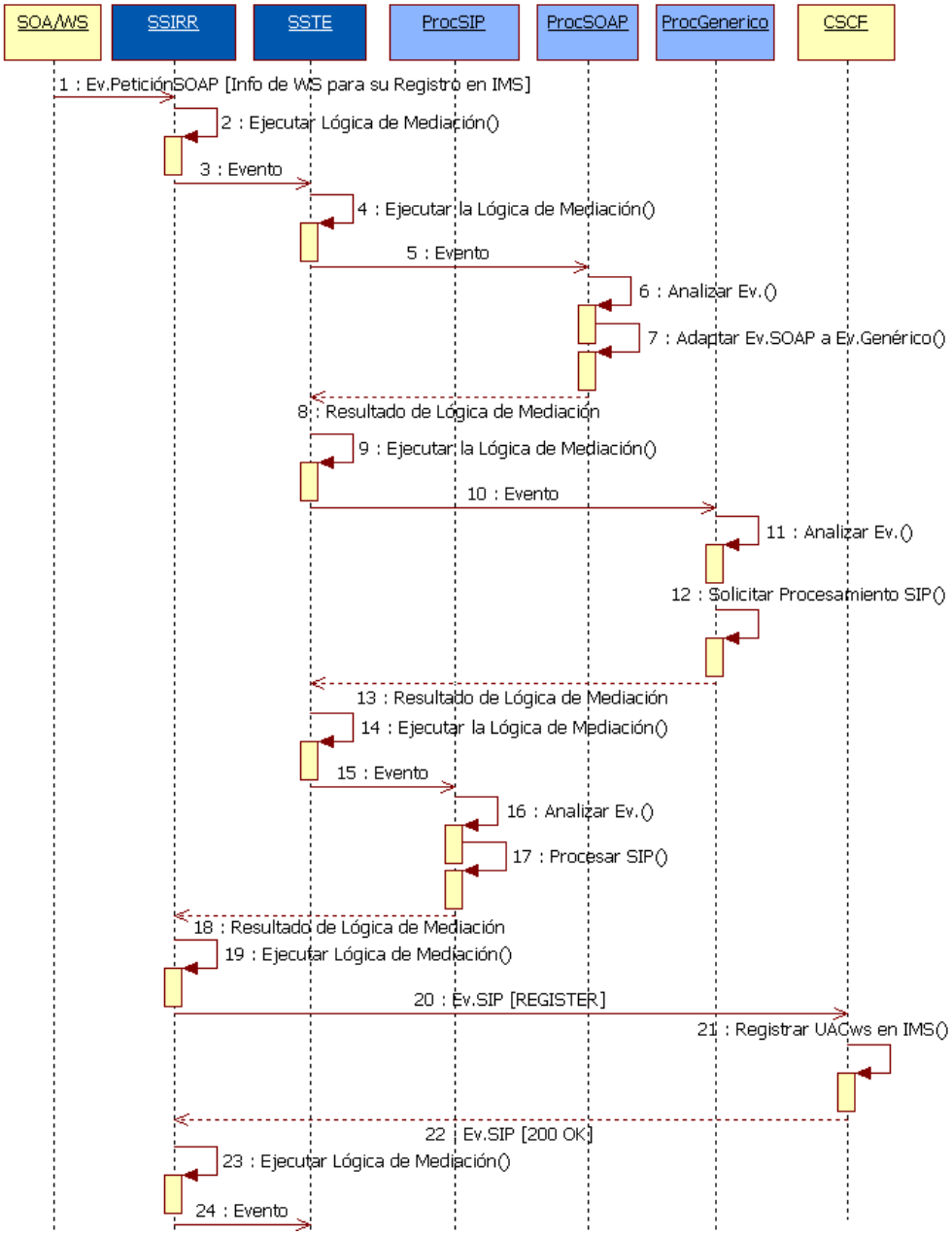
- El componente SOA/WS, específicamente el Proveedor de WS, envía al SSIRR un Evento de Petición SOAP con la información necesaria para el registro del WS en IMS. (*Paso 1*)
- El SSIRR primero ejecuta la lógica de mediación necesaria tanto para la adaptación del evento SOAP a un evento genérico, como para el inicio de su contexto, y finalmente envía el evento al SSTE.(*Pasos 2 y 3*)

- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento de petición del tipo SOAP, decide dirigirlo al SSMIDD. (*Pasos 4 y 5*)
- El Procesador SOAP del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento SOAP recibido y la información que contiene, tras lo cual, se determina que se trata de una solicitud que contiene una operación de petición para el registro de un WS en IMS. Por otra parte, se analiza el contexto del proceso donde se está realizando la transacción del evento y también se almacenan y/o recuperan los datos persistentes relacionados con el proceso particular. Como en este caso se trata de un evento de tipo SOAP, a través del cual se requiere de la mediación en el registro de un WS en IMS, se determina que la tarea que se realizará a continuación es la adaptación del evento de tipo SOAP a un evento de tipo genérico, donde este último es el elemento de mediación entre las solicitudes de los protocolos de comunicación SIP y SOAP. Durante el proceso de adaptación se utilizan los datos enviados a través del evento SOAP, correspondientes básicamente al *Service Port* del WS y al UAS SIP que se desea registrar en IMS, para la generación de la solicitud de procesamiento SIP que será enviada posteriormente al Procesador de este tipo de protocolo. Adicionalmente, en el proceso de adaptación se almacenan y/o recuperan los datos persistentes requeridos. Finalmente, el resultado de la mediación es enviado al SSTE. (*Pasos 6-8*)
- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento genérico, decide dirigirlo al SSMIDD. (*Pasos 9 y 10*)
- El Procesador Genérico del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento Genérico recibido y la información que contiene, tras lo cual se determina que se trata de una solicitud que contiene una operación de petición para el registro de un WS en IMS. Por otra parte, se analiza el contexto del proceso donde se está realizando la transacción del evento y también se almacenan y/o recuperan los datos persistentes relacionados con el proceso particular. Como en este caso se trata de un evento de tipo genérico, a través del cual se requiere de la mediación en el registro de un WS en IMS, se determina que la tarea que se realizará a continuación es la generación de una solicitud de procesamiento SIP, cuyo destino es el Procesador de este tipo de protocolo. Finalmente, el resultado de la mediación es enviado al SSTE.(*Pasos 11-13*)
- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento de Petición del tipo genérico, decide dirigirlo al SSMIDD.(*Pasos 14 y 15*)
- El Procesador SIP del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento Genérico recibido y la información que contiene, tras lo cual se determina que se trata de una solicitud que contiene una operación de petición para el registro de un WS en IMS. Por otra parte, se analiza el contexto del proceso donde se está realizando la transacción del evento y también se almacenan y/o recuperan los datos persistentes, correspondientes al *Service Port* del WS y al UAS SIP que se desea registrar en IMS. Como en este caso se trata de un evento de tipo Genérico con la solicitud de un procesamiento SIP, se determina que

la tarea que se realizará a continuación es la generación de un evento de registro SIP, teniendo en cuenta los datos recibidos a través del evento Genérico, provenientes de la solicitud SOAP original. Finalmente, el resultado de la mediación es enviado al SSIRR. (*Pasos 16-18*)

- El SSIRR recibe el resultado del proceso de mediación, representado por un evento genérico del tipo SIP, lo adapta a un evento específico de Petición SIP del tipo REGISTER, además gestiona el contexto del proceso que involucra dicho evento, y finalmente lo entrega al componente CSCF de la red IMS.(*Pasos 19 y 20*)
- El Núcleo IMS, específicamente el CSCF, ejecuta el proceso de registro del WS en IMS para que su uso esté disponible en los procesos de creación de los servicios convergentes IMS/SOA, y envía al SSIRR un evento de respuesta SIP del tipo 200 OK (En el diagrama no se tienen en cuenta los procesos de señalización SIP de autorización requeridos por la red IMS).(*Pasos 21 y 22*)
- El SSIRR ejecuta la lógica de mediación necesaria tanto para la adaptación del evento SIP a un evento genérico, como para la gestión del contexto del proceso que involucra dicho evento, y finalmente lo envía al SSTE. (*Pasos 23 y 24*)
- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento de Respuesta del tipo SIP, decide dirigirlo al SSMIDD. (*Pasos 25 y 26*)
- El Procesador SIP del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento SIP recibido y la información que contiene, tras lo cual se determina que se trata de una respuesta de tipo SIP exitosa, o de 200 OK, en el registro de un UAC de WS en IMS. Por otra parte, se analiza el contexto del proceso donde se está realizando la transacción del evento y también se almacenan y/o recuperan los datos persistentes relacionados con el proceso particular. Como en este caso se trata de un evento de tipo SIP, a través del cual se confirma el registro exitoso de un WS en IMS, se determina que la tarea que se realizará a continuación es la adaptación del evento de tipo SIP a un evento de tipo genérico, donde este último es el elemento de mediación entre las solicitudes de los protocolos de comunicación SIP y SOAP. Finalmente, el resultado de la mediación es enviado al SSTE.(*Pasos 27-29*)
- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué Subsistema está interesado en el evento. En este caso, como se trata de un evento de Petición del tipo Genérico, decide dirigirlo al SSMIDD.(*Pasos 30 y 31*)
- El Procesador Genérico del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento Genérico recibido y la información que contiene, tras lo cual se determina que se trata de una solicitud que contiene una operación de respuesta exitosa en el registro de un WS en IMS. Por otra parte, se analiza el contexto del proceso donde se está realizando la transacción del evento y también se almacenan y/o recuperan los datos persistentes relacionados con el proceso particular. Como en este caso se trata de un evento de tipo genérico, a través del cual se requiere de la mediación en la respuesta exitosa al registro de un WS en IMS, se determina que la tarea que se realizará a continuación es la generación de una solicitud de procesamiento SOAP, cuyo destino es el procesador de este tipo de protocolo. Finalmente, el resultado de la mediación es enviado al SSTE.(*Pasos 32-34*)

- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento de Petición del tipo Genérico, decide dirigirlo al SSMIDD.(*Pasos 35 y 36*)
- El Procesador SOAP del Subsistema recibe el evento y lleva a cabo la primera tarea que consiste en su análisis. Por una parte, dentro de este análisis se explora el tipo de evento genérico recibido y la información que contiene, tras lo cual se determina que se trata de una solicitud de un procesamiento SOAP para la generación de respuesta exitosa al registro de un WS en IMS. Finalmente, el resultado de la mediación es enviado al SSIRR. (*Pasos 37-39*)
- El SSIRR recibe el resultado del proceso de mediación, representado por un evento genérico, lo adapta a un evento específico de respuesta SOAP, además gestiona el contexto del proceso que involucra dicho evento, y finalmente lo entrega al proveedor de WS del componente SOA/WS. (*Pasos 40 y 41*)



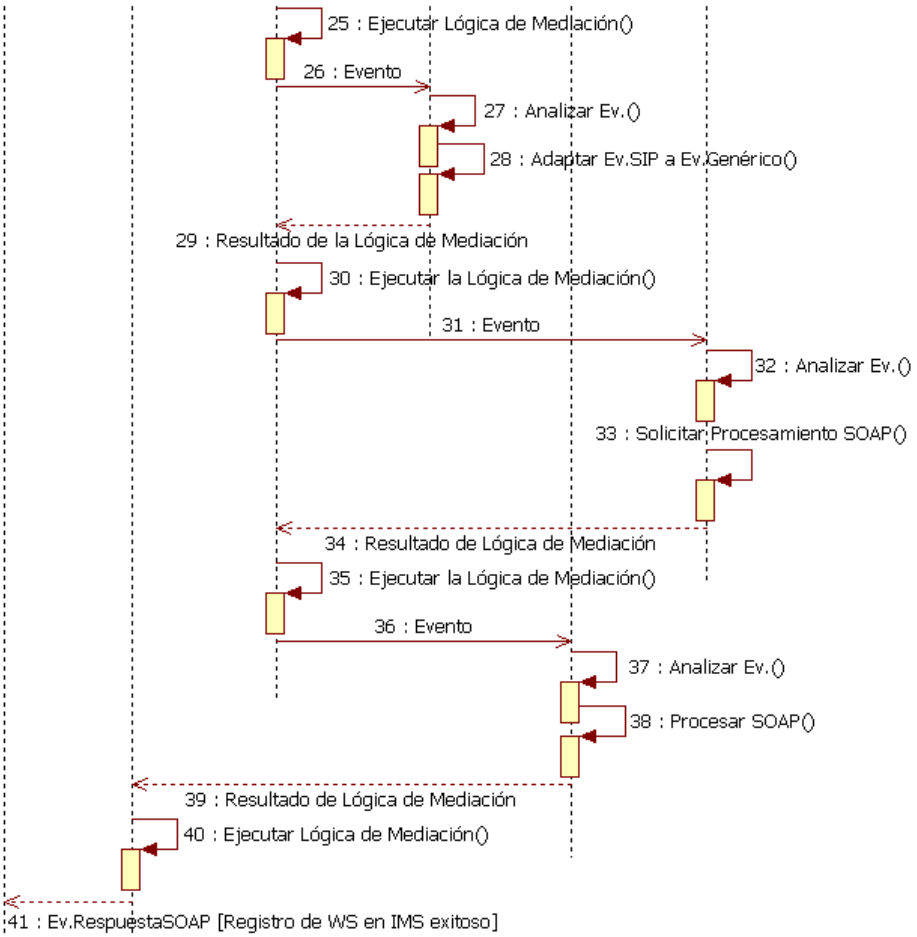


Figura 15. Diagrama de Interacción de Componentes del Subsistema de Mediación SIP/SOAP. Interacción entre los Subsistemas de Interfaces de Recursos de Red, de Transmisión de Eventos y los Procesadores SIP y SOAP

3.2.1.3.4. SSREGS

Es el subsistema encargado de la mediación en el Registro de las descripciones de los Servicios IMS y Web en el modelo de registro unificado basado en UDDI de la Lógica de Mediación, por lo tanto es una funcionalidad de la Lógica de Mediación orientada a los proveedores de Servicios IMS y Web.

Este Subsistema provee su funcionalidad al SSMIDD, por lo cual ocupa una escala inferior a la de este, dentro de la jerarquía de Subsistemas de la Lógica de mediación. La interacción entre el SSREGS y el SSMIDD es a través del SSTE.

Este subsistema se encarga de:

- Recibir y procesar Eventos Genéricos provenientes del SSMIDD para la publicación y consulta de las descripciones de los Servicios IMS y Web.
- Administrar el registro de servicios basado en UDDI de la Lógica de Mediación.
- Generar eventos de Tipo Genérico, resultado de los procesos de análisis y procesamiento, en respuesta a las solicitudes recibidas para la publicación o consulta de Servicios. El destino de estos eventos es el SSMIDD.

Para cumplir con estas funcionalidades el subsistema se subdivide en los componentes mostrados en la Figura 16.

Diagrama de componentes del Subsistema

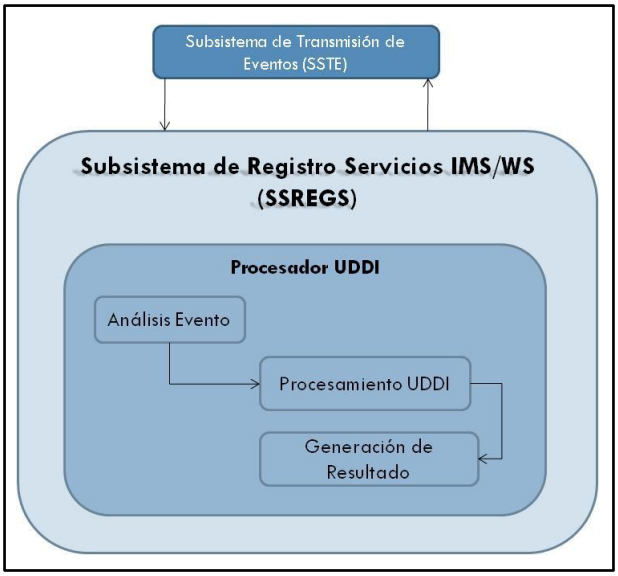


Figura 16. Diagrama de Componentes del Subsistema de Registro de Servicios (IMS/WS)

Componentes del Subsistema

- **Procesador UDDI**: El SSREGS recibe eventos de tipo genérico, provenientes del SSTE, y los entrega al Procesador UDDI para que inicialmente se encargue de su análisis, a continuación del procesamiento UDDI requerido (por ejemplo, publicación, consulta, actualización de la información de descripción del servicio), y finalmente la generación de los resultados del procesamiento. La primera tarea de este componente es el análisis del evento recibido, la cual consiste, por una parte, en la exploración del evento, y, por otra, en el análisis del contexto del proceso donde se está realizando la transacción del evento y en el almacenamiento y/o recuperación de los datos persistentes relacionados con el proceso particular. El resultado de este análisis es la ejecución de la tarea de procesamiento UDDI, la cual puede consistir en la publicación, actualización o consulta de la descripción de un Servicio IMS o Web. Dicha tarea es llevada a cabo a través de la gestión del registro de servicios basado en UDDI de la Lógica de Mediación. La última tarea de este procesador es la generación del resultado del procesamiento como un evento genérico que es entregado al SSTE para su posterior re-envío hacia el SSMIDD.

El SSREGS no posee interfaces internas ya que solo cuenta con un componente el cual se encarga de toda su funcionalidad.

Diagrama de interacción de componentes

En la Figura 17 se muestra la interacción entre el Proveedor de WS y los Subsistemas de Interfaces de Recursos de Red, de Transmisión de Eventos, de Mediación SIP/SOAP y de Registro de Servicios IMS/WS.

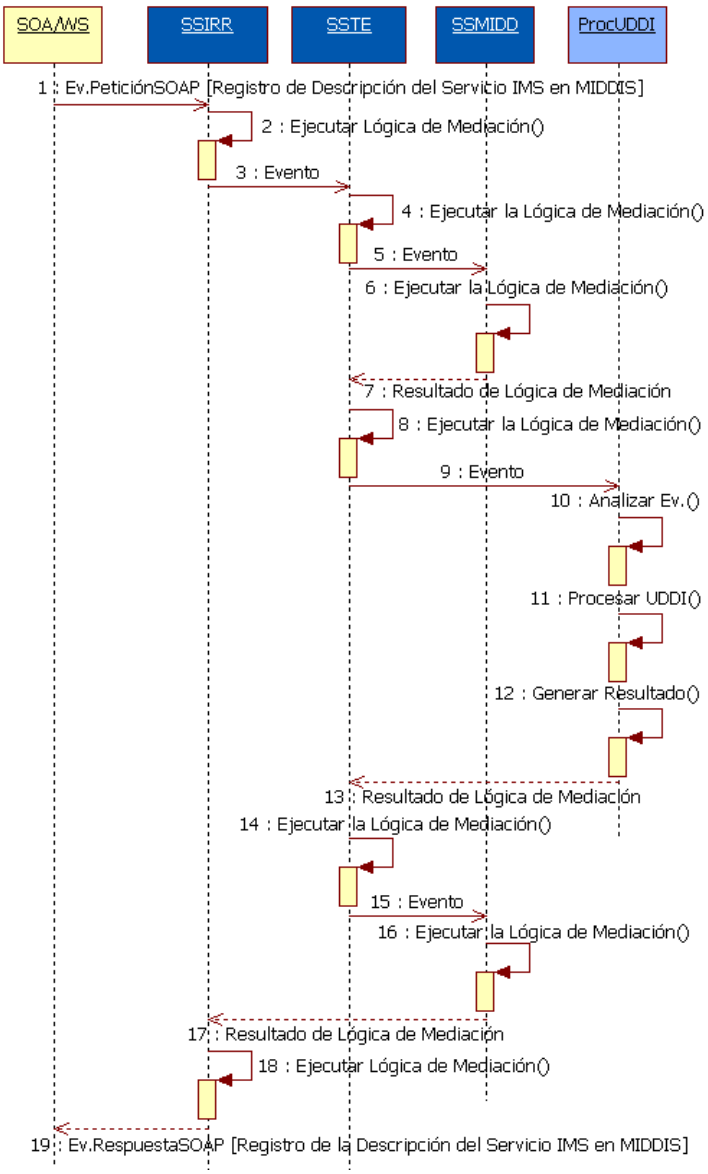


Figura 17. Diagrama de Interacción de Componentes del Subsistema de Registro de Servicios IMS/WS. Interacción entre los Subsistemas de Interfaces de Recursos de Red, de Transmisión de Eventos, de Mediación SIP/SOAP, y el Procesador UDDI

A continuación se describe el Diagrama de interacción de componentes del Subsistema de Registro de Servicios (IMS/WS), para el proceso de Registro de un Servicio IMS en el modelo de registro basado en UDDI de la Lógica de Mediación.

Descripción

- El componente SOA/WS, específicamente el Proveedor de WS, envía al SSIRR un Evento de Petición SOAP con los datos necesarios para el proceso de registro de la descripción de un Servicio IMS en el registro UDDI de la Lógica de mediación. (*Paso 1*)
- El SSIRR ejecuta la lógica de mediación necesaria tanto para la adaptación del evento SOAP a un evento genérico, como para el inicio del contexto del proceso que involucra dicho evento, y finalmente lo envía al SSTE. (*Pasos 2 y 3*)

- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento de Petición del tipo SOAP, decide dirigirlo al SSMIDD. (*Pasos 4 y 5*)
- El Procesador SOAP del SSMIDD recibe el evento, y realiza el análisis respectivo determinando que se trata de una solicitud que contiene una operación de petición para el registro de un Servicio IMS en el registro UDDI de la Lógica de Mediación. A continuación, adapta el evento de tipo SOAP a un evento de tipo Genérico. Durante el proceso de adaptación se utilizan los datos enviados a través del evento SOAP, correspondientes a la información de descripción del Servicio IMS, para la generación de la solicitud de procesamiento UDDI que será enviada al Procesador de este tipo de datos, dentro del SSREGS. Adicionalmente, en el proceso de adaptación se almacenan y/o recuperan los datos persistentes requeridos. Finalmente, el resultado de la mediación es enviado al SSTE. (*Pasos 6 y 7*)
- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué Subsistema está interesado en el evento. En este caso, como se trata de un evento Genérico para procesamiento UDDI, decide dirigirlo al SSREGS.(*Pasos 8 y 9*)
- El Procesador UDDI del Subsistema recibe el evento, y realiza el análisis respectivo explorando el tipo de evento Genérico y la información que contiene, tras lo cual se determina que se trata de una solicitud de publicación de la descripción de un Servicio IMS en el registro UDDI de la Lógica de Mediación. A continuación, en la tarea de ejecución del procesamiento UDDI, se obtienen, del evento recibido, los datos correspondientes a:
  - La compañía desarrolladora del servicio y su información de contacto (businessEntityName, businessEntityDescription, businessEntityContactPhone, businessEntityContactEmail),
  - Información técnica del Servicio (businessServiceName, businessServiceDescription, bindingTemplateLocation, bindingTemplateArchitecture, bindingTemplateTModel). (Oasis, 2002)

  Luego se realiza la inserción de dichos datos en el registro UDDI de la Lógica de Mediación. Finalmente, el resultado de la publicación es retornado al SSMIDD a través del SSTE. (*Pasos 10-13*)

- El SSTE ejecuta la lógica de mediación necesaria para la determinación de qué subsistema está interesado en el evento. En este caso, como se trata de un evento genérico para procesamiento SOAP, decide dirigirlo al SSMIDD. (*Pasos 14 y 15*)
- El Procesador Genérico del SSMIDD recibe el evento y realiza el análisis respectivo. Por una parte, dentro de este análisis se explora el tipo de evento genérico recibido y la información que contiene, tras lo cual se determina que se trata de una solicitud de generación de respuesta exitosa por la publicación de la descripción de un Servicio IMS en el registro UDDI de la Lógica de Mediación. Como en este caso se trata de un evento de tipo genérico, a través del cual se requiere de la mediación en la generación de una respuesta SOAP exitosa, se determina que se debe generar una solicitud de procesamiento SOAP. A través del SSTE se envía dicha solicitud hacia el Procesador SOAP del SSMIDD, que recibe el evento, lo analiza y determina que se trata de una solicitud de generación de respuesta exitosa al registro de un WS en IMS. Finalmente, el resultado de la mediación es enviado al SSIRR. (*Pasos 16 y 17*)

- El SSIRR recibe el resultado del proceso de mediación, representado por un evento genérico, lo adapta a un evento específico de Respuesta SOAP, además gestiona el contexto del proceso que involucra dicho evento, y finalmente lo entrega al Proveedor de WS del componente SOA/WS.(*Pasos 18 y 19*)

3.2.1.3.5. SSMED

Este subsistema realiza la mediación entre el plano de medios de IMS/SIP y el de SOA/WS. Entre sus funciones están:

- Recibir y procesar Eventos de medios de la infraestructura de red subyacente, a través del SSTE.
- Enviar Eventos de medios a la infraestructura de red subyacente, a través del SSIRR.
- Adaptar los formatos del contenido Web de los servicios SOA/WS a los formatos y protocolos de medios soportados por la red IMS y sus clientes.
- Enviar los Resultados de la adaptación (datos formateados) al SSIRR, para su posterior envío al Cliente IMS.

Para cumplir con estas funcionalidades el subsistema se subdivide en los componentes mostrados en la Figura 18.

Diagrama de componentes del Subsistema

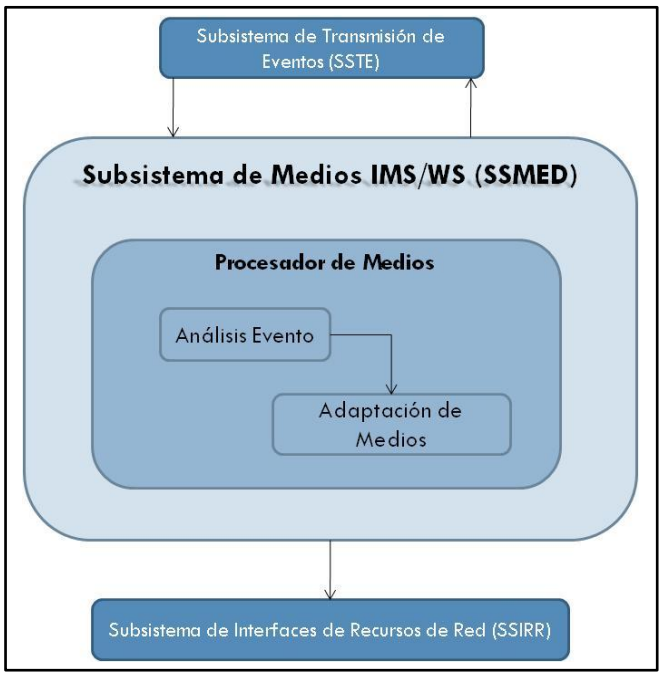


Figura 18. Diagrama de Componentes del Subsistema de Medios (IMS/WS)

Componentes del Subsistema

- **Procesador de Medios**: El SSMED recibe eventos de medios, provenientes del SSTE, y entrega dichos eventos al Procesador de Medios para que inicialmente se encargue de su análisis, a continuación de la adaptación requerida, y finalmente de la generación del

evento de medios resultado del procesamiento. La primera tarea de este componente es el análisis del evento recibido, la cual consiste, por una parte, en la exploración del evento, y, por otra parte, en el análisis del contexto del proceso donde se está realizando la transacción del evento y en el almacenamiento y/o recuperación de los datos persistentes relacionados con el proceso particular. El resultado de este análisis es la ejecución de la tarea de adaptación de los formatos del contenido Web, de los servicios SOA/WS, a los formatos y protocolos de medios soportados por la red IMS, los cuales son finalmente enviados al SSIRR que, a través de su Adaptador de Medios, los envía al Cliente IMS de la red subyacente.

El SSMED no posee interfaces internas ya que solo cuenta con un componente, el cual se encarga de toda su funcionalidad.

Capítulo 4

IMPLEMENTACIÓN DE REFERENCIA

En este capítulo se describe la implementación de la arquitectura. Inicialmente se define el alcance de la misma y las decisiones de diseño consideradas. Posteriormente, se amplía la descripción de la implementación detallando la forma como se instanciaron los Subsistemas que conforman la Lógica de Mediación, y sus respectivos componentes, así como también los elementos externos a ella, que conforman la infraestructura de red subyacente a la cual proporciona sus funcionalidades.

4.1. Implementación de Referencia de MIDDIS

4.1.1. Alcance

La implementación de referencia persigue dos objetivos principales, asociados a los objetivos de esta tesis de maestría:

1. Instanciar la Arquitectura de Referencia para proporcionar las funcionalidades necesarias para la interacción entre servicios basados en SOA e IMS con el fin de crear Servicios Convergentes SOA/IMS.
2. Desarrollar un piloto de Servicio Convergente SOA/IMS que consuma capacidades de un Servicio SOA sobre una red IP y capacidades de un servicio de telecomunicaciones sobre una red IMS, el cual permita realizar la evaluación experimental de la efectividad y eficiencia del enfoque adoptado en la arquitectura de referencia.

4.1.1.1. Características

De acuerdo a los requisitos se definen las siguientes características para la implementación de referencia:

- **Soporte al protocolo SIP-3GPP de IMS**: la implementación de referencia debe dar soporte al protocolo SIP (UAS y UAC SIP) en la Lógica de Mediación, así como también debe dar soporte a este tipo de señalización en el entorno de la red subyacente la cual está conformada por: el Cliente IMS, el Núcleo IMS y el Servicio basado en IMS.
- **Soporte al protocolo SOAP**: la implementación de referencia debe dar soporte al protocolo SOAP en la Lógica de Mediación, así como también en el componente SOA/WS, externo a la misma, para proporcionar el medio necesario para el transporte bidireccional de los mensajes de información del WS que hace parte del Servicio Convergente.
- **Mediación SIP/SOAP**: la implementación de referencia debe soportar la interacción, a nivel de señalización y control de servicios, entre SIP (IMS) y SOAP (SOA/WS), para permitir el registro, inicio, mantenimiento, y terminación de sesiones IMS (a través de SIP), así como también el acceso a las funcionalidades de los WS (a través de SOAP), y a su característica de publicación y consulta de servicios a través de UDDI.

- **Soporte a la creación de Servicios Convergentes IMS/SOA**: la implementación de referencia debe soportar el fácil acceso a las funcionalidades de los WS desde el entorno IMS, para facilitar la creación rápida de Servicios Convergentes basados en IMS y en SOA.

4.1.1.2. Restricciones

En este apartado se listan las consideraciones que se tuvieron en cuenta para realizar la instanciación de la arquitectura para la implementación de referencia del presente trabajo:

1. En la implementación del diseño propuesto se ha excluido la instanciación de la extensión WS-Context de la especificación de WS, y solamente se define su utilización para el soporte del concepto de sesión en el entorno SOA.
2. En 3.2.1.3.5 se define el Subsistema de Medios IMS/SOA para la adaptación de los medios Web a los medios soportados por IMS. No obstante, debido a la complejidad y robustez de este Subsistema solamente se describe su funcionalidad dentro MIDDIS, así mismo se excluye su implementación ya que no es necesario para realizar la validación del mediador creado.

4.1.2. Arquitectura

La Figura 19 muestra la Arquitectura de la Implementación de Referencia, la cual es una instanciación de la Arquitectura de Referencia planteada en el capítulo anterior.

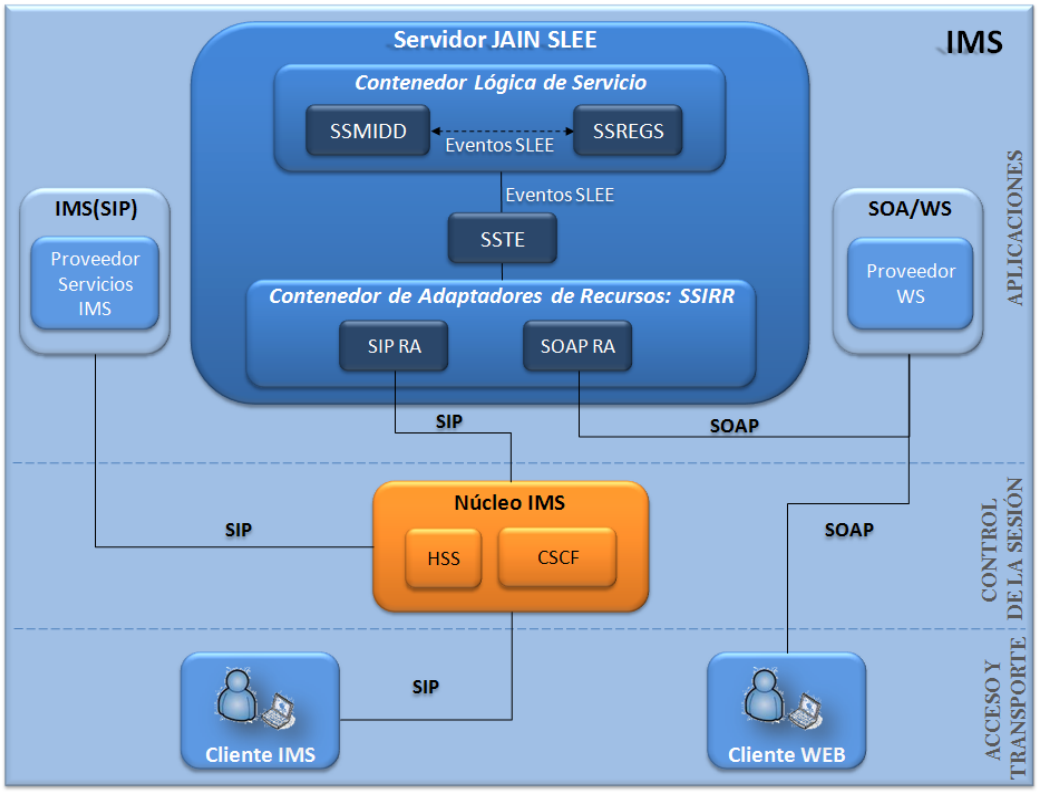


Figura 19. Arquitectura de la Implementación de Referencia

En la Arquitectura de la Implementación de Referencia (Figura 19), MIDDIS, alojada en el Servidor JAIN SLEE, permite al entorno SOA/WS acceder a las funcionalidades de IMS y

proporcionar el acceso a sus propias capacidades desde la red de Telecomunicaciones. Específicamente, el Proveedor de WS permite al Cliente Web registrarse en IMS, y puede suministrar tanto el acceso a las funcionalidades de los WS como también realizar las operaciones de gestión de las descripciones de los Servicios IMS y Web. Así mismo, MIDDIS permite al Cliente IMS, desde el entorno de las Telecomunicaciones y a través del Núcleo de IMS, iniciar, mantener y terminar sesiones SIP-IMS con un WS, así como también acceder a su funcionalidad. Los anteriores constituyen los principales aportes que realiza el presente trabajo de grado de maestría.

En el Anexo B se incluyen todos los detalles de las herramientas y tecnologías para la implementación y ejecución de MIDDIS.

4.1.2.1. Subsistema de Interfaces de Recursos de Red

La implementación del SSIRR de MIDDIS se realizó a través de un elemento constitutivo de la arquitectura de JSLEE: el plano de RA, debido a la correspondencia en características y funcionalidades requeridas e implementadas entre estos elementos. La arquitectura del RA permite la interconexión de JSLEE con el mundo exterior, para lo cual proporciona, por ejemplo, las interfaces para las pilas de protocolos de comunicaciones, servicios de directorio o sistemas externos (tales como el rating y la facturación o *billing*).

4.1.2.2. Subsistema de Transmisión de Eventos

La implementación del SSTE de MIDDIS se realizó a través de un elemento que hace parte del marco de trabajo de JSLEE: el Enrutador de Eventos, debido también a la correspondencia en características y funcionalidades requeridas e implementadas entre estos elementos. El entorno de JSLEE invoca los SBB de acuerdo a un modelo estandarizado del ciclo de vida, similar al de los EJB. Durante las transacciones el entorno de ejecución asegura y gestiona el procesamiento de eventos y la invocación del marco de trabajo; al hacerlo, el servidor de aplicaciones de JSLEE permanece en un estado definido y consistente, incluso en caso de fallos.

4.1.2.3. Subsistema de Mediación IMS/SOA y Subsistema de Registro de Servicios IMS-WS

Para la creación de la lógica de servicio de MIDDIS, conformada por los subsistemas restantes: SSMIDD y SSREGS, se utilizó la FSM Tool, versión 0.8.5 (OpenCloud, 2009b), en conjunto con XDoclet versión 2.1.0, ambas de OpenCloud, las cuales son herramientas livianas, orientadas a simplificar la creación de servicios para Rhino SLEE.

Los módulos de XDoclet se utilizan para la generación automática de los descriptores de despliegue para los componentes que se despliegan en el OpenCloud Rhino, en base a etiquetas especiales en los archivos fuente de Java. Desde la perspectiva del desarrollador, la FSM Tool se asegura que los artefactos de la especificación y la implementación permanezcan completamente precisos y sincronizados durante todo el desarrollo y el mantenimiento del ciclo de vida de cada componente. Además, éste no tiene que codificar a mano ni mantener una jerarquía compleja de las Máquinas de Estados Finitos (Finite State Machine, FSM) que conforman el sistema.

El modelado de servicios a partir de FSM inicia con el diseño de los diagramas de secuencia para todos los procesos que la aplicación debe ejecutar. Una vez se hayan diseñado se puede empezar a crear un conjunto de máquinas de estados que manejen los flujos de llamada definidos en los diagramas. Se puede pensar en cada diagrama de secuencia como un camino a través de la máquina de estados.

Las FSM de los servicios que proporciona MIDDIS se definieron formalmente en un Lenguaje de Dominio Específico (Domain-Specific Language, DSL) basado en texto. Los principales pasos seguidos para el desarrollo de cada uno de los servicios de MIDDIS por medio de la FSM Tool fueron:

1. Se escribió la especificación de cada FSM, de manera que describiera completamente el comportamiento de los protocolos de comunicaciones SIP y SOAP. El modelado FSM de un protocolo de comunicaciones puede ser descrito como una tupla $(\Sigma,\Gamma,S,s_0,\delta,\omega)$, donde Σ es el alfabeto de entrada, Γ es el alfabeto de salida, S es un conjunto no vacío y finito de estados, s_0 es el estado inicial (pertenece a S), δ es la función de la transición δ: $S\times\Sigma$-> S, ω es la función de salida ω: $S\times\Sigma$->Γ (Basicevic, y otros, 2010).

   A partir de la definición anterior, la especificación de una FSM consiste en un archivo de texto en el que se definen los estados, las acciones, las transiciones, las entradas y las salidas. La sintáxis del VFSM DSL incluye:

   - Definiciones y descripciones de los estados.
   - Entradas, salidas, y acciones de ingreso asociadas con cada estado junto con las condiciones que se deben cumplir antes de la ejecución.
   - Transiciones, con las condiciones que se deben cumplir antes de realizar la transición
   - Un diccionario de entrada (`inputdictionary`), en el que se declaran todas las entradas (eventos de entrada) a las que hacen referencia las definiciones de los estados.
   - Un diccionario de salida (`outputdictionary`), en el que se declaran todas las salidas a las que hacen referencia las definiciones de los estados.

   De acuerdo a los servicios proporcionados por el SSMIDD y el SSREGS, y específicamente a los flujos de eventos específicos de cada protocolo de comunicación en los que basan su funcionalidad, se definieron 4 FSM, 2 para cada protocolo de comunicaciones dependiendo del tipo de servicios provistos.

El primer grupo de especificaciones de FSM da soporte a los procesos de Registro de un WS en IMS, y a la Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI de MIDDIS. Este grupo de FSM constituyen un aporte fundamental del presente trabajo ya que en la investigación documental realizada no se encontró un esquema como el propuesto y menos su respectiva implementación.

- **SoapMiddStateMachine.vfsm**

La máquina de estados incluye (Figura 20):

- *Estados*: `INITIAL`, `REGISTERED`, `ERROR`, y `FINAL`
- *Eventos*: `soapRequest`, `register`, `successful`, `proccesing` y `error`

- Una acción de entrada para el estado `INITIAL`: `checkSoapRequest`
- Una acción de entrada para el estado `REGISTERED`: `checkSoapRequest`

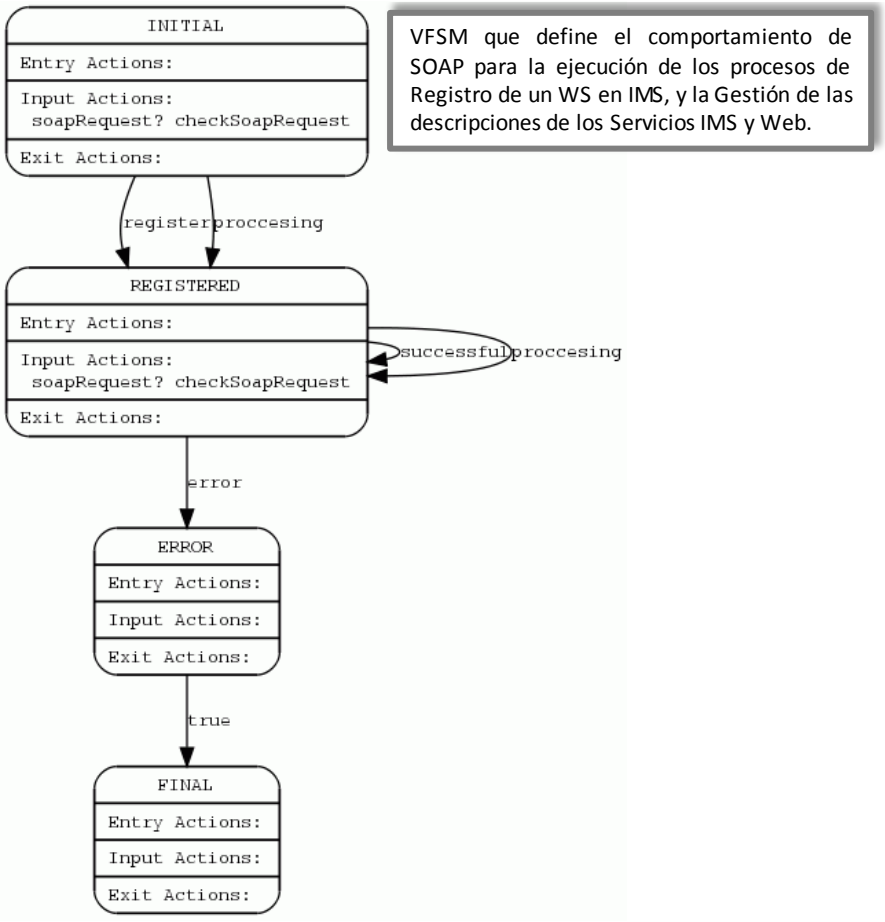


Figura 20. SoapMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `soapRequest`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapRequest` en la cual se analiza la solicitud SOAP recibida.
- Si se fijan las entradas `register` o `proccesing`, la máquina ejecuta la transición hacia el estado `REGISTERED`. Las entradas `register` y `proccesing` son fijadas para identificar los procesos de registro de un WS en IMS y de gestión UDDI en MIDDIS, respectivamente.

En el estado `REGISTERED`:

- Cuando la máquina recibe el evento `soapRequest`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapRequest` en la cual se analiza la solicitud SOAP recibida.
- La máquina ejecuta la transición hacia el mismo estado y hacia el estado ERROR al fijarse las entradas `successful|proccesing` y `error`, respectivamente. Las entradas `successful` y `proccesing` son fijadas para identificar los procesos de envío de respuesta SOAP exitosa, desde el SLEE hacia la red subyacente, y de gestión UDDI en

MIDDIS, respectivamente. La entrada `error` es fijada cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

Cuando la máquina de estados llega al estado `ERROR`, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado `FINAL`.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

- **SipMiddStateMachine.vfsm**

La máquina de estados incluye (Figura 21):

- *Estados*: `INITIAL`, `REGISTER`, `ERROR`, y `FINAL`
- *Eventos*: `clientErrorResponse`, `timeOut`, `registering`, `error`, `okResponse`, `successRegister` y `endSession`
- Acciones de entrada para el estado `INITIAL`: `clientErrorResponse`, `timeOut`
- Acciones de entrada para el estado `REGISTER`: `okResponse`, `timeOut`

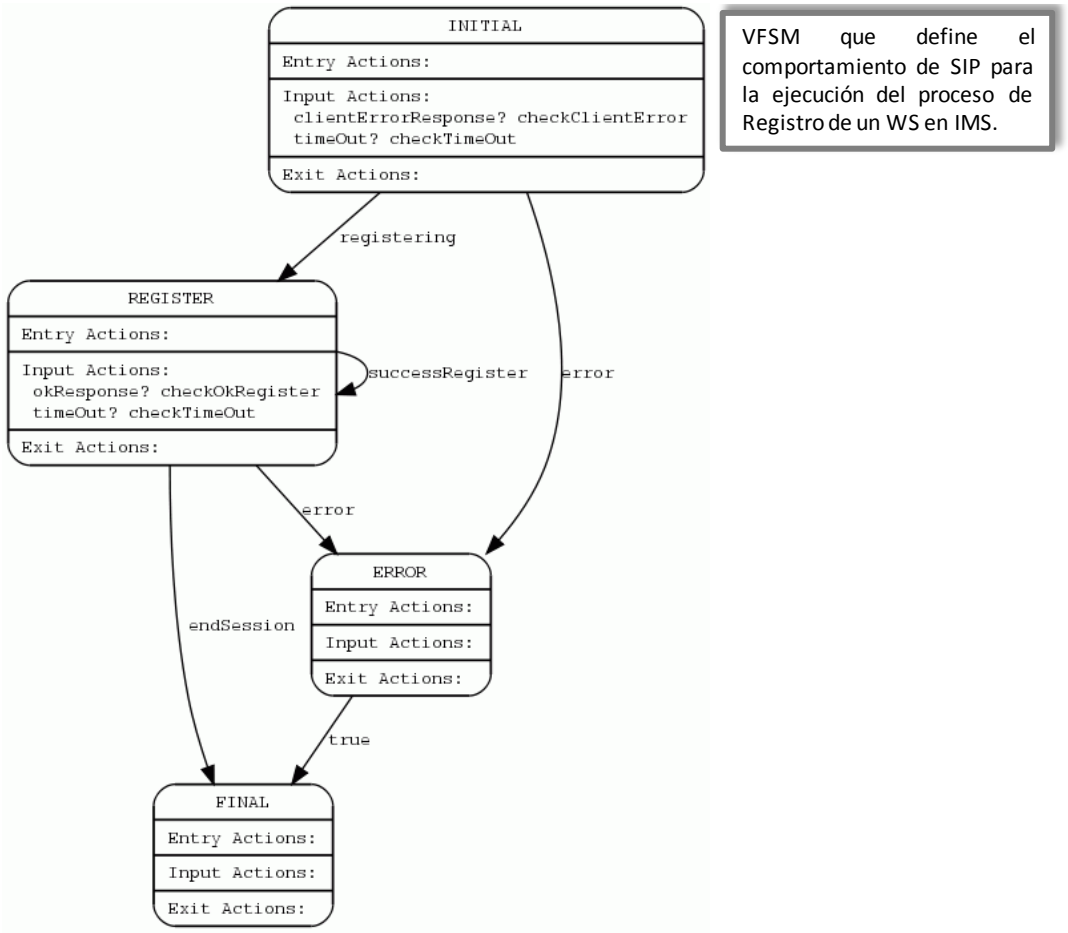


Figura 21. SipMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `clientErrorResponse`, debido a la llegada de un evento de respuesta SIP de tipo error del cliente o `401 UNAUTHORIZED` al JSLEE, ésta

ejecuta la acción de entrada `checkClientError` en la cual se analiza la respuesta SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `registering` y `error`, la máquina ejecuta la transición hacia el estado `REGISTER` y `ERROR`, respectivamente. Las entradas `registering` y `error` son fijadas para identificar los procesos de generación de una petición SIP de registro hacia IMS y de ocurrencia de un error, respectivamente.

En el estado `REGISTER`:

- Cuando la máquina recibe el evento `okResponse`, debido a la llegada de un evento de respuesta SIP de tipo `200 OK` al JSLEE, ésta ejecuta la acción de entrada `checkOkRegister` en la cual se procesa la confirmación exitosa a la petición de registro del WS en IMS.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- La máquina ejecuta la transición hacia el mismo estado, hacia el estado `ERROR`, y hacia el estado `FINAL`, al fijarse las entradas `successRegister`, `error`, y `endSession`, respectivamente. La entrada `successRegister` se fija para identificar el proceso de confirmación de registro exitoso de un WS en IMS; la entrada `error` es fijada cuando se produce algún error durante uno de los procesos que ejecuta la lógica del servicio; y la entrada `endSession` se fija cuando se requiere terminar la ejecución de la máquina.

Cuando la máquina de estados llega al estado `ERROR`, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado `FINAL`.

En el estado`FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

El segundo grupo de especificaciones de FSM da soporte a los procesos de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS. Aportes fundamentales del presente trabajo.

- **SipUASMiddStateMachine.vfsm**

La máquina de estados incluye (Figura 22):

- *Estados*: `INITIAL`, `CONNECTING`, `CONVERGED`, `ERROR`, y `FINAL`
- *Eventos*: `soapRequest`, `register`, `successful`, `proccesing` y `error`
- Las acciones de entrada para el estado `INITIAL`: `inviteRequest`, `timeOut`, `messageRequest`, `byeRequest`
- Las acciones de entrada para el estado `CONNECTING`: `ackRequest`, `timeOut`
- Una acción de entrada para el estado `CONVERGED`: `timeOut`

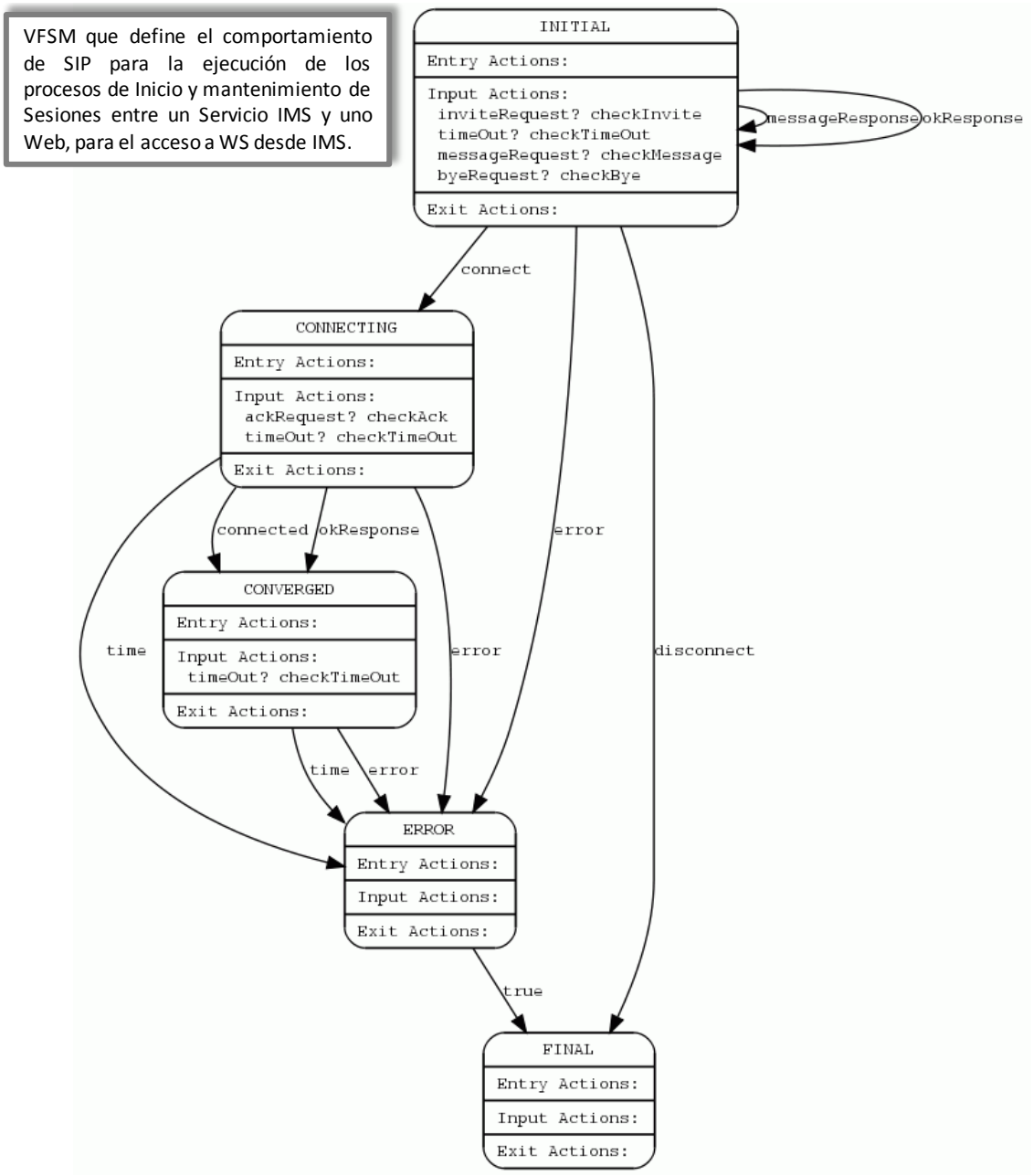


Figura 22. SipUASMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `inviteRequest`, debido a la llegada de un evento SIP de tipo `INVITE` al JSLEE, ésta ejecuta la acción de entrada `checkInvite` en la cual se analiza la petición SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Cuando la máquina recibe el evento `messageRequest`, debido a la llegada de un evento SIP de tipo `MESSAGE` al JSLEE, ésta ejecuta la acción de entrada `checkMessage` en la cual se procesa la invocación de un WS desde IMS.

- Cuando la máquina recibe el evento `byeRequest`, debido a la llegada de un evento SIP de tipo `BYE` al JSLEE, ésta ejecuta la acción de entrada `checkBye` en la cual se procesa la petición de finalización de sesión entre el UA IMS y el WS.
- Si se fijan las entradas `messageResponse`o`okResponse`, la máquina ejecuta la transición hacia el estado `INITIAL`; si se fijan las entradas `connect`, `disconnect`, y `error`, la máquina ejecuta la transición hacia los estados `CONNECTING`, `FINAL`, y `ERROR`, respectivamente. La entrada `messageResponse` se fija para identificar el proceso de generación de una petición SIP de tipo `MESSAGE` que contenga el resultado de la invocación al WS; la entrada `okResponse` se fija para identificar el proceso de recepción de una respuesta SIP de tipo `200 OK`; la entrada `connect` se fija para identificar que se ha iniciado el proceso de establecimiento de una sesión entre el UA IMS y el WS; la entrada `disconnect` se fija para identificar el proceso de finalización de una sesión entre el UA IMS y el WS; la entrada `error` se fija cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

En el estado `CONNECTING`:

- Cuando la máquina recibe el evento `ackRequest`, debido a la llegada de un evento SIP de tipo `ACK` al JSLEE, ésta ejecuta la acción de entrada `checkAck` en la cual se analiza la petición SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `connected` o `okResponse`, la máquina ejecuta la transición hacia el estado `CONVERGED`; si se fijan las entradas `time` o `error`, la máquina ejecuta la transición hacia el estado `ERROR`. La entrada `connected` se fija para identificar el proceso confirmación del establecimiento exitoso de sesión entre el UA IMS y el WS; la entrada `okResponse` se fija para identificar el proceso de envío de una respuesta SIP de tipo `200 OK`; la entrada `time` se fija para identificar la ocurrencia de un evento de temporizador durante alguno de los procesos que ejecuta la lógica del servicio; la entrada `error` se fija cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

En el estado `CONVERGED`:

- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `error` o `time`, la máquina ejecuta la transición hacia el estado `ERROR`. Las entradas `time` y `error` se fijan para identificar la ocurrencia de un evento de temporizador y de un error, respectivamente, durante alguno de los procesos que ejecuta la lógica del servicio.

Cuando la máquina de estados llega al estado `ERROR`, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado `FINAL`.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

- **SoapUASMiddStateMachine.vfsm**

La máquina de estados incluye (Figura 23):

- *Estados*: `INITIAL`, `CONVERGED`, y `FINAL`
- *Eventos*: `soapResponse`, `generatingSession`, `soapOkInviteResponse`, `invoking`y`endSession`
- Una acción de entrada para el estado `INITIAL`: `soapResponse`
- Una acción de entrada para el estado `CONVERGED`: `soapResponse`

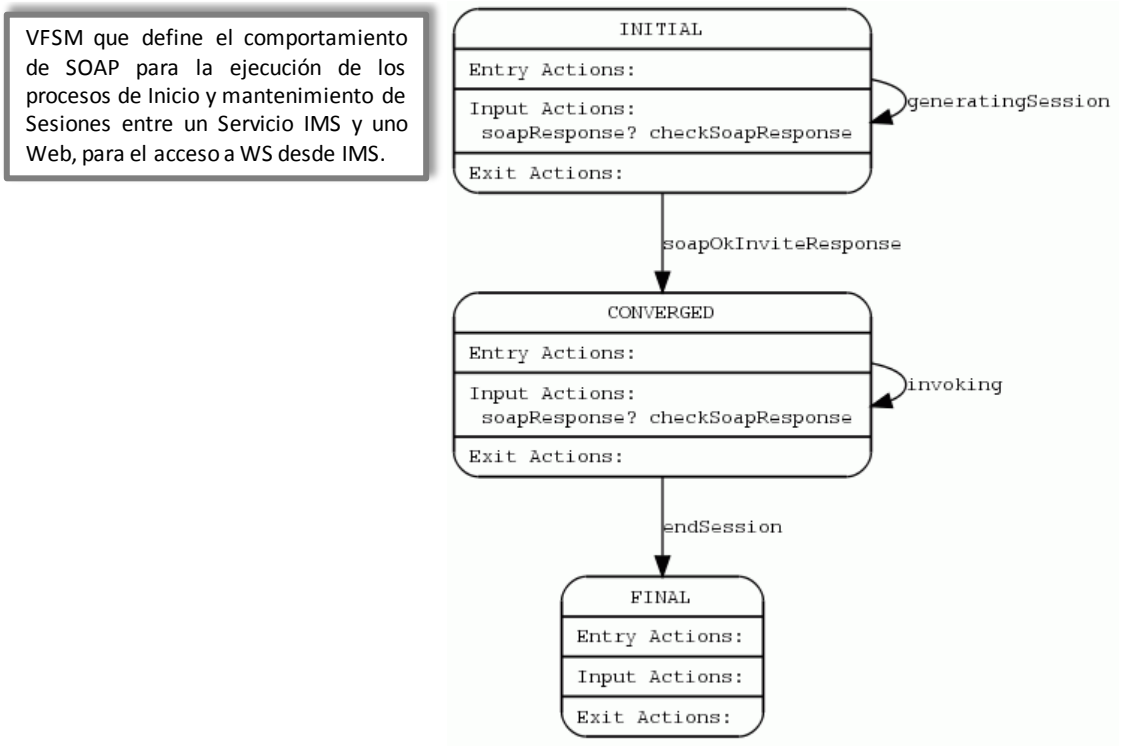


Figura 23. SoapUASMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `soapResponse`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapResponse` en la cual se procesa la respuesta SOAP recibida.
- Si se fijan las entradas `generatingSession` y `soapOkInviteResponse` la máquina ejecuta la transición hacia el mismo estado y hacia el estado `CONVERGED`, respectivamente. Las entradas `generatingSession` y `soapOkInviteResponse` son fijadas para identificar los procesos de generación de una petición de inicio de sesión desde el SLEE hacia el WS y de confirmación exitosa al inicio de sesión entre el UA IMS y el WS, respectivamente.

En el estado `CONVERGED`:

- Cuando la máquina recibe el evento `soapResponse`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapResponse` en la cual se procesa la respuesta SOAP recibida.

- La máquina ejecuta la transición hacia el mismo estado y hacia el estado `FINAL` al fijarse las entradas `invoking` y `endSession`, respectivamente. Las entradas `invoking` y `endSession` son fijadas para identificar los procesos de envío de una petición SOAP para la invocación de un WS, desde el SLEE hacia la red subyacente, y de terminación de sesión entre el UA IMS y el WS, respectivamente.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

2. La FSM Tool utiliza la especificación de la máquina de estados para generar automáticamente una clase Java del SBB de la FSM. En este proceso se generaron las clases: *SipMiddStateMachineSbb.java*, *SipUASMiddStateMachineSbb.java*, *SoapMiddStateMachineSbb.java*, y *SoapUASMiddStateMachineSbb.java*. Se recomienda la utilización de la herramienta XDoclet de OpenCloud para generar los descriptores de despliegue para los SBB.

3. La especificación VFSM define el comportamiento abstracto del componente. Para hacer que funcione, se debió extender la clase Java FSM SBB generada, en el paso 2, con una clase SBB que implementará las acciones definidas en la especificación de la FSM. Esta nueva clase SBB debe: mapear los eventos recibidos a las acciones de entrada, y ejecutar la máquina de estados; e implementar los métodos de acción que la clase generada declara de manera abstracta. Los desarrolladores de JAIN SLEE están acostumbrados a escribir la lógica en los métodos manejadores de eventos del SBB. La FSM Tool cambia este estilo, e implementa todo el código de la lógica del servicio en los métodos de las acciones, y sólo se utilizan los métodos manejadores de eventos para el mapeo de estos últimos a las entradas de la máquina de estados. Por lo tanto:
   - El SBB recibe un evento, a través del RA correspondiente, en el método manejador de evento respectivo,
   - se notifica a la máquina de estados la recepción de una entrada,
   - la VFSM redirige el procesamiento de dicho evento al método de acción especificado.

4. Finalmente, se compiló, empaquetó y desplegó a MIDDIS utilizando los métodos estándar para el despliegue completo de los SBB. El ejecutable del software creado consiste en los archivos fuente, las librerías, la unidad desplegable, y el descriptor del despliegue del servicio.

- **Diseño e Implementación de la Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI**

Los procesos de publicación, consulta y modificación de las descripciones de los Servicios IMS y Web en MIDDIS se basan en el modelo UDDI de los WS (Oasis, 2002), y su uso en este contexto constituye uno de los principales aportes del presente trabajo de grado de maestría.

Los datos necesarios para el registro de la descripción de un servicio en el UDDI de MIDDIS son:

```
<uddiInfo>
<businessEntityName>Entity Name</businessEntityName>
<businessEntityDescription>Entity
Description</businessEntityDescription>
```

```
<businessEntityContactPhone> Entity
ContactPhone</businessEntityContactPhone>
<businessEntityContactEmail>Entity Contact Email
</businessEntityContactEmail>
<businessServiceName>Service Name</businessServiceName>
<businessServiceDescription>Service
Description</businessServiceDescription>
<bindingTemplateLocation>Template Location</bindingTemplateLocation>
<bindingTemplateArchitecture>Template
Architecture</bindingTemplateArchitecture>
<bindingTemplateTModel>Template TModel</bindingTemplateTModel>
</uddiInfo>
```

- **businessEntityName**: Compañía que desarrolla el servicio.
- **businessEntityDescription**: Descripción de la compañía que desarrolla el servicio.
- **businessEntityContactPhone**: Información de contacto de la compañía.
- **businessEntityContactEmail**: Información de contacto de la compañía.
- **businessServiceName**: Información técnica del servicio.
- **businessServiceDescription**: Información que describe técnicamente el servicio.
- **bindingTemplateLocation**: Hace parte de la plantilla que almacena la descripción técnica del servicio concerniente a su localización. Es decir, la dirección o localización donde se puede encontrar la implementación actual del servicio.
- **bindingTemplateArchitecture**: Hace parte de la plantilla que almacena la descripción técnica del servicio concerniente a la descripción de su arquitectura.
- **bindingTemplateTModel**: Hace parte de la plantilla que almacena las especificaciones técnicas que describen la naturaleza de un servicio. Si se trata de la descripción de un Servicio IMS se almacena la ruta a los javadoc del servicio dentro de la etiqueta `<overviewsip>`, y la WSDL del WS dentro de la etiqueta `<overviewsoap>`. En el caso de un WS, se incluye solamente la WSDL dentro de la etiqueta `<overviewsoap>`.

En MIDDIS, la implementación de la publicación, consulta y modificación de las descripciones de los servicios basada en UDDI se realizó por medio de los Perfiles de JAIN SLEE, como se especifica en la versión 1.1 del API (Capítulo 10 de (Sun Microsystems y OpenCloud, 2009)).

Los Perfiles son objetos que contienen datos requeridos por un componente (por ejemplo un SBB) para realizar sus funciones. Entre dichos datos se pueden encontrar datos de configuración o datos del suscriptor. La Especificación de un Perfil describe el esquema de las Tablas del Perfil. En dicha especificación se define el conjunto de atributos para cada Tabla del Perfil.

La especificación 1.1 del SLEE define 2 tipos de consultas: estática y dinámica. Toda consulta estática que sea requerida se declara en la especificación del perfil; mientras, que las consultas dinámicas pueden ser declaradas después de que se haya desplegado una Especificación de un Perfil.

Los métodos de los Perfiles creados en JSLEE permiten realizar las operaciones de creación, eliminación, y consulta de los datos de las descripciones de los Servicios IMS o Web en un perfil UDDI de MIDDIS. Algunos de ellos se describen a continuación:

```
//Creación de un perfil:
MiddUDDIProfileCMP profile =
(MiddUDDIProfileCMP)getProfileTable().create( profileName );
//Adición de los Atributos del perfil:
profile.setBusinessEName( entityname );
            profile.setBusinessEDescription( entitydesc );
…
//Eliminación de un perfil:
//El objeto TABLE_NAME, de tipo cadena, representa el nombre de la
//tabla de perfiles UDDI dentro de MIDDIS
getProfileFacility().getProfileTable( TABLE_NAME ).remove(
profileName );

//Consulta del dato businessEntityName de un perfil:
MiddUDDIProfileCMP profile =
(MiddUDDIProfileCMP)getProfileTable().find( profileName );
String response = "";
response = "businessEntityName: " + profile.getBusinessEName();
```

Es importante tener en cuenta que para poder utilizar los perfiles dentro del JSLEE debe existir la tabla de perfiles correspondiente. En MIDDIS dicha tabla es creada por medio de la tarea de gestión del SLEE denominada `createprofiletable`.

Para la ejecución de la Gestión, basada en UDDI, de las descripciones de los Servicios IMS y Web se utilizan peticiones de tipo SOAP. Por una parte, el registro o modificación de la descripción de un servicio se ejecuta por medio de un documento XML, que representa el *SOAP Envelope* de la petición SOAP, en cuyo cuerpo, o `Body`, se incluye la operación, en este caso `setUddi`, el nombre del perfil dentro de la etiqueta `<webServiceName>`, y los parámetros correspondientes a la descripción misma del servicio, de acuerdo al modelo UDDI de MIDDIS. El nombre del perfil incluido dentro de esta petición SOAP permitirá determinar primero la existencia del mismo dentro del registro UDDI de MIDDIS, y posterior a este análisis permitirá seleccionar la operación a ejecutar, que puede ser la adición de un nuevo perfil o la actualización de uno existente. A continuación se muestra un ejemplo de este tipo de documento XML, para el registro de la descripción de un Servicio IMS en MIDDIS, utilizando UDDI:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
<soa>
<operation>setUddi</operation>
<webServiceName>WSfootwearshopService</webServiceName>
<uddiInfo>
<businessEntityName>Git</businessEntityName>
<businessEntityDescription>Grupo de Ingenieria Telematica Unicauca
</businessEntityDescription>
<businessEntityContactPhone>+57 (2) 8209800 ext
2127</businessEntityContactPhone>
<businessEntityContactEmail>git@unicauca.edu.co</businessEntityContac
tEmail>
<businessServiceName>IMSWSConvergedClient</businessServiceName>
<businessServiceDescription>Servicio Convergente  IMS/WS de acceso a
Wsfootwearshop</businessServiceDescription>
<bindingTemplateLocation>sip:open-
ims.test:7060</bindingTemplateLocation>
<bindingTemplateArchitecture>http://192.168.0.1:8080/ConvergedService
```

```
s/IMSWSConvergedClient_architecture.html</bindingTemplateArchitecture
>
<bindingTemplateTModel><overviewsip>middis/doc/javadoc</overviewsip>
<overviewsoap>http://192.168.0.1:8080/ConvergedServices/WSfootwearsho
pService?WSDL</overviewsoap>
</bindingTemplateTModel>
</uddiInfo>
</soa>
</Body>
</Envelope>
```

Por otra parte, el proceso de consulta de la descripción de un servicio se ejecuta también por medio de un documento XML, que representa el *SOAP Envelope* de la petición SOAP, en cuyo cuerpo, o `Body`, se incluyec la operación, en este caso `getUddi`, y una etiqueta `<webServiceName>` que puede contener el nombre del perfil a consultar, o el carácter "–", por medio del cual se consulta la lista de los nombres de los perfiles registrados actualmente en MIDDIS. A continuación, se muestra un ejemplo de este tipo de documento XML:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
      <soa>
            <operation>getUddi</operation>
            <webServiceName>WSfootwearshopService</webServiceName>
      </soa>
</Body>
</Envelope>
```

4.1.3. Modelo de Despliegue

En la Figura 24, que muestra el Diagrama de Implantación de la Arquitectura diseñada para la Implementación de referencia, la Arquitectura de servicios IMS comprende las funciones lógicas, estratificadas por niveles, encargadas del Acceso y Transporte, el Control de la sesión, y las Aplicaciones; además, se detallan los componentes incluidos en la Implementación de referencia y su distribución en los nodos empleados.

- **Nivel de Acceso y Transporte**: Cliente IMS y Web, desarrollados en Java y soportados por el JDK.
- **Nivel de Control de la sesión**: Núcleo IMS, constituido por el HSS y los CSCF, y simulado a través del Open IMS Core (Fraunhofer FOKUS, 2009).
- **Nivel de Aplicaciones**: De tipo IMS/SIP, con el Servicio IMS, y de tipo SOA/WS, con el Servicio Web. Este último desarrollado en Java y desplegado en un servidor de aplicaciones Glassfish. Adicionalmente, en el nivel de Aplicaciones de IMS se encuentra MIDDIS desplegado en un Servidor JAIN SLEE Rhino, que contiene los módulos de adaptación a recursos externos (SIP RA, SOAP RA) y los bloques de lógica de servicio que implementan sus Subsistemas (SIP SBB, SOAP SBB, y Middleware SBB).

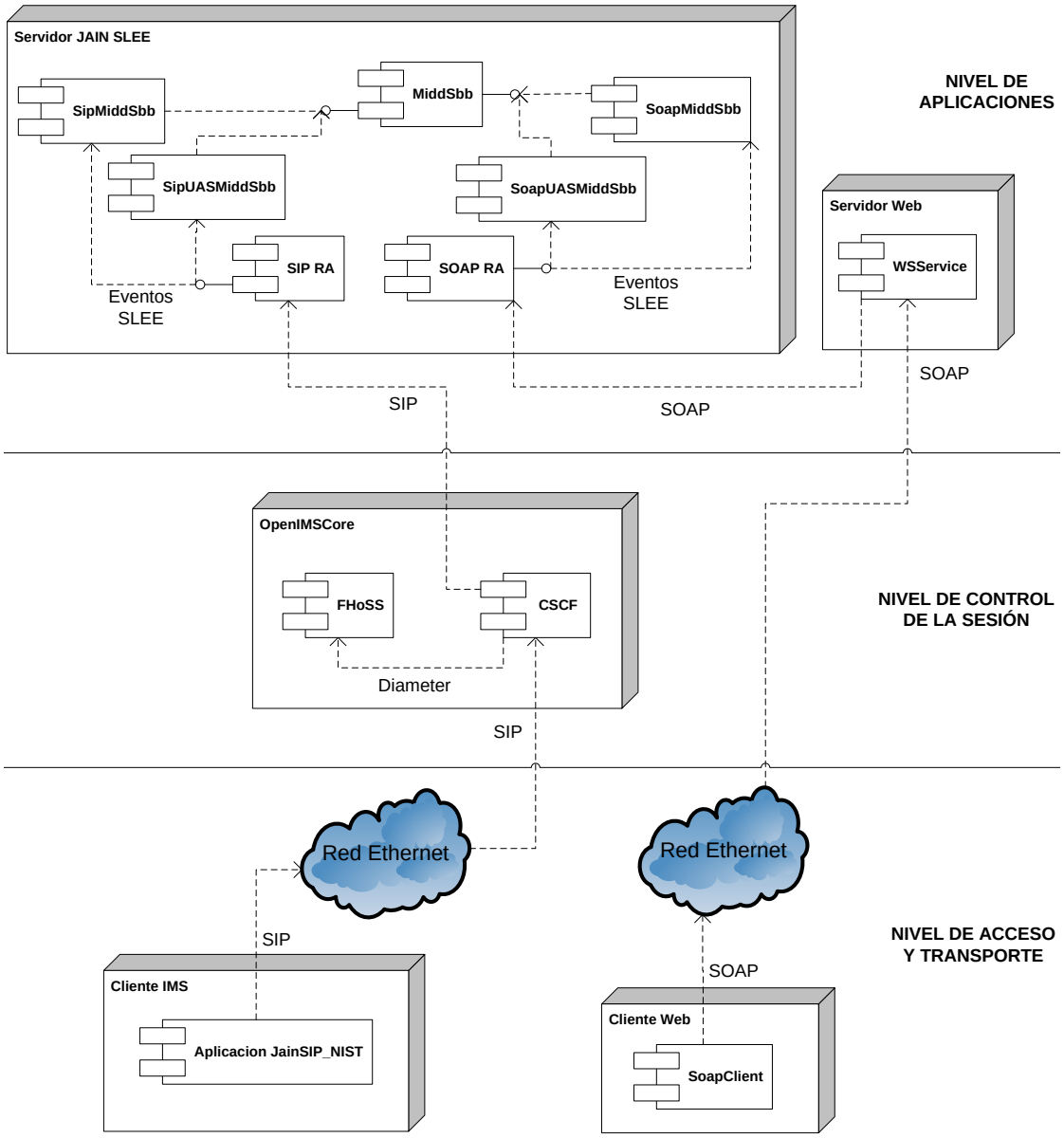


Figura 24. Modelo de Implantación de la Implementación de Referencia

En el Servidor JAIN SLEE se despliegan los RA para SIP y SOAP. Por una parte, MIDDIS se comunica con la red subyacente, correspondiente al Open IMS Core y al Cliente IMS, a través del SIP RA. Por otra parte, MIDDIS se comunica con la red subyacente, correspondiente a la infraestructura del WS desplegada en el Servidor Web, a través del SOAP RA. De igual manera, en el Servidor JAIN SLEE se detallan los componentes SBB desplegados, en los cuales se encuentran distribuidas las funcionalidades del SSMIDD y el SSREGS que conforman a MIDDIS.

El Cliente Web utiliza un Cliente SOAP, desplegado en el Servidor JAIN SLEE, que por una parte genera peticiones SOAP y las envía a MIDDIS, y por otra parte procesa las respuestas SOAP que MIDDIS le entrega, durante los procesos de registro del WS en la red Telecomunicaciones, y la gestión de las descripciones de los servicios IMS y Web. El WS, desplegado en el Servidor Web, provee el conjunto de métodos necesarios para el inicio, mantenimiento y terminación de sesiones con un Cliente IMS, a través de MIDDIS, así como también los métodos específicos del WS.

El Cliente IMS utiliza una aplicación basada en JAIN SIP y NIST para registrarse en la red Telecomunicaciones, que es simulada a través del Open IMS Core, y de esta manera habilitar el inicio, mantenimiento, y terminación de sesiones, así como el acceso a las funcionalidades del WS, a través de MIDDIS.

Las características del hardware y del software utilizado en la implementación de referencia de MIDDIS y del piloto de servicio convergente se describen a continuación:

- **MIDDIS**: se implantó sobre Rhino SLEE SDK versión 2.1_03 de OpenCloud (OpenCloud, 2010), con las versiones 2.2_06 del SIP RA, y 2.1 del SOAP RA. La instalación se realizó en el OS Ubuntu 9.10 Server, y la Máquina Virtual Java (JVM, Java Virtual Machine) versión 1.6.0_20. El motor de base de datos instalado para este ambiente de ejecución fue PostgreSQL 8.4.3, al cual se accedió a través de JDBC.
- **P-CSCF, I-CSCF, C-CSCF y HSS**: conforman el Núcleo de IMS, y es proporcionado por el Open IMS Core, instalado en un en un computador con procesador Intel Core 2 1,66GHz, 1.000 MB de RAM bajo el S.O. Linux Ubuntu 9.10 server.
- **Servidor Web**: GlassFish v3, instalado en un en un computador con procesador Intel Pentium 4 1,70GHz, 1.000 MB de RAM bajo el S.O. Windows XP.
- **Cliente IMS**: Cliente JAIN SIP-NIST, ejecutado en un computador con procesador Intel Core 2 1,66GHz, 1.000 MB de RAM bajo el S.O. Linux Ubuntu 9.10 server.
- **Cliente Web**: Cliente Java con procesamiento de SOAP, ejecutado en un computador con procesador Intel Core 2 1,66GHz, 1.000 MB de RAM bajo el S.O. Linux Ubuntu 9.10 server.

4.2. Evaluación de la Arquitectura de Referencia

Inicialmente en esta sección se describen los criterios que permitieron la evaluación del desempeño de MIDDIS. Luego se presenta el piloto de Servicio Convergente creado para ilustrar cómo se realiza el acceso a las capacidades de un Servicio SOA, sobre una red IP, y a las capacidades de un servicio de telecomunicaciones, sobre una red IMS, a través del mediador diseñado e implementado. Posteriormente se describen las pruebas de señalización y de rendimiento realizadas, junto con los resultados y el análisis respectivo. Finalmente, se describe cómo se ejecutan la publicación, consulta y modificación de las descripciones de los Servicios IMS y Web en MIDDIS.

4.2.1. Criterios de Evaluación

4.2.1.1. Indicadores para la evaluación del desempeño de SIP

En la RFC 6076 (Malas, y otros, 2011) se plantean las siguientes métricas o indicadores para la evaluación del desempeño de SIP.

- **Retardo en la Petición de Registro (Registration Request Delay, RRD)**

Este indicador, ilustrado en la Figura 25, es la medida del retardo en la respuesta a un UA para una petición SIP de tipo `REGISTER`. Se calcula utilizando la siguiente fórmula:

```
RRD = Tiempo de la Respuesta Final – Tiempo de la petición REGISTER
```

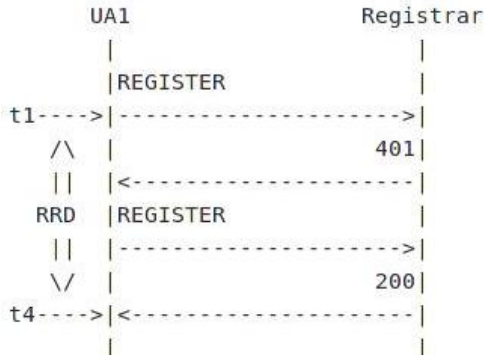


Figura 25. Retardo en la Petición de Registro

- **Retardo en la Petición de Inicio de Sesión (Session Request Delay, SRD)**

Este indicador, ilustrado en la Figura 26, se calcula utilizando la siguiente fórmula:

```
SRD = Tiempo de la Respuesta Indicativa del Estado – Tiempo de la
petición INVITE
```

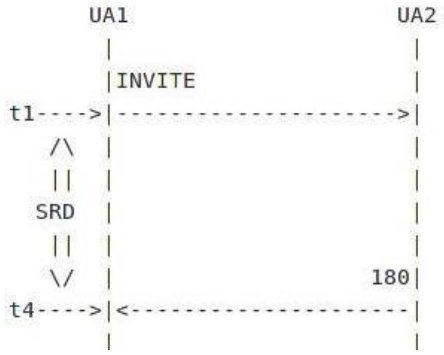


Figura 26. Retardo en la Petición de Inicio de Sesión

En un intento exitoso de petición, el SDR se define como el intervalo de tiempo desde cuando el UA de origen envía el mensaje INVITE inicial, conteniendo la información necesaria, hacia el elemento mediador o UA de destino, hasta la primera respuesta provisional recibida, indicando un estado visual o audible de la petición inicial de configuración de la sesión. En SIP, el mensaje que indica el estado es un mensaje provisional, distinto del tipo 100 o Trying, recibido en respuesta a una petición INVITE. En algunos casos no se reciben este tipo de respuestas sino mensajes 200 OK como primer mensaje de estado. En estos casos, el mensaje 200 OK sería utilizado para calcular el intervalo. En la mayoría de circunstancias, este indicador se basa en la recepción de un mensaje diferente al tipo 100.

- **Retardo en la Finalización de la Sesión (Session Disconnect Delay, SDD)**

Este indicador, ilustrado en la Figura 27, se calcula utilizando la siguiente fórmula:

```
SDD = Tiempo de la Respuesta 2XX o Timeout – Tiempo de finalización
del mensaje BYE
```

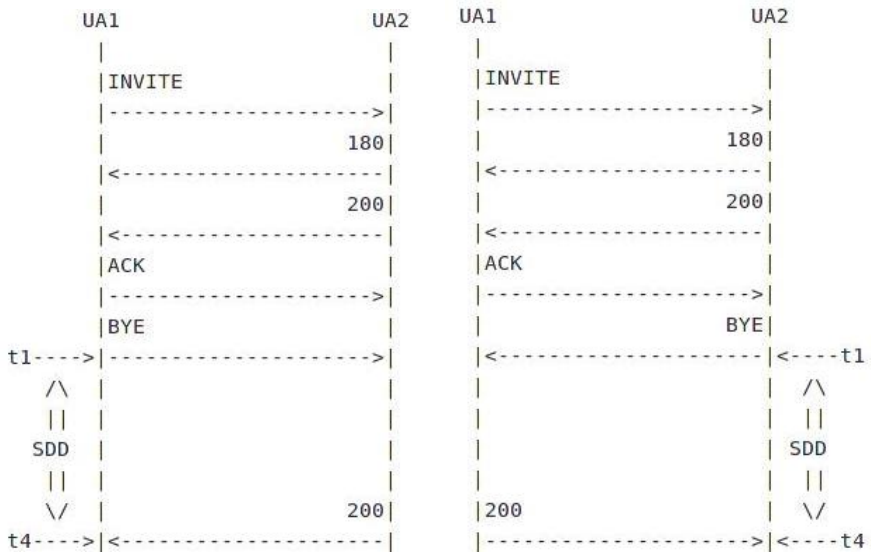


Figura 27. Retardo en la Finalización de la Sesión

4.2.1.2. Criterios para la evaluación del desempeño de sitios Web

Debido a que los servicios proporcionados por MIDDIS no solamente están dirigidos hacia el entorno de telecomunicaciones, sino también hacia el entorno Web, la evaluación y clasificación de los resultados de las pruebas del mediador también se realizó teniendo en cuenta el conjunto de reglas que determinan los criterios de tiempos de respuesta para aplicaciones en el entorno Web, las cuales están directamente relacionadas con las características cognitivas de los humanos (Joines, y otros, 2002):

- Retardo menor a 0.1 segundos: los usuarios no notan este retardo, por lo tanto se clasifica como un tiempo de respuesta óptimo.
- Retardo menor a 1 segundo: no interrumpe el flujo del pensamiento del usuario, pero hace manifiesto un retardo, por lo tanto se clasifica como un tiempo de respuesta bueno.
- Retardo menor a 10 segundos: los usuarios aún esperarán la respuesta, pero el retardo es notorio. Por tanto, se clasifica como un tiempo de respuesta aceptable.
- Retardo mayor a 10 segundos: los usuarios pierden la concentración y no continúan esperando la respuesta del sistema. Por tanto, se clasifica como un tiempo de respuesta deficiente.

4.2.2. Evaluando Invocación de un Servicio Web desde IMS

Para ilustrar el funcionamiento de la arquitectura de referencia de MIDDIS se realizó la implementación de la invocación de un sitio Web de comercio electrónico, de venta de zapatos, desde un servicio IMS. El WS creado, `Wsfootwearshop`, permite a los usuarios la consulta, selección de artículos, y posterior compra de los mismos.

El modelo de implementación general incluye: al Cliente IMS, usuario final del servicio; al CSCF, elemento que soporta los servicios IMS del usuario de telecomunicaciones, y por lo tanto el encargado de la invocación del `Wsfootwearshop`; a MIDDIS, que proporciona la mediación entre las interacciones con las capacidades del terminal del Cliente IMS y con el WS; y al componente SOA/WS, que provee el `Wsfootwearshop`.

En el diagrama de secuencia que se muestra en la Figura 28, se describe el proceso de envío de una orden de compra desde un Cliente IMS hacia el `Wsfootwearshop`, a través de MIDDIS. Una vez registrados tanto el terminal del Cliente IMS, como el `WSfootwearshop` en la red IMS, e iniciada la sesión entre los mismos, el Servicio Convergente IMS/WS provee al Cliente IMS el acceso al WS por medio de la invocación de las operaciones de consulta, selección de artículos, y procesamiento de orden de compra, enviadas a MIDDIS como eventos de petición SIP, a través del CSCF.

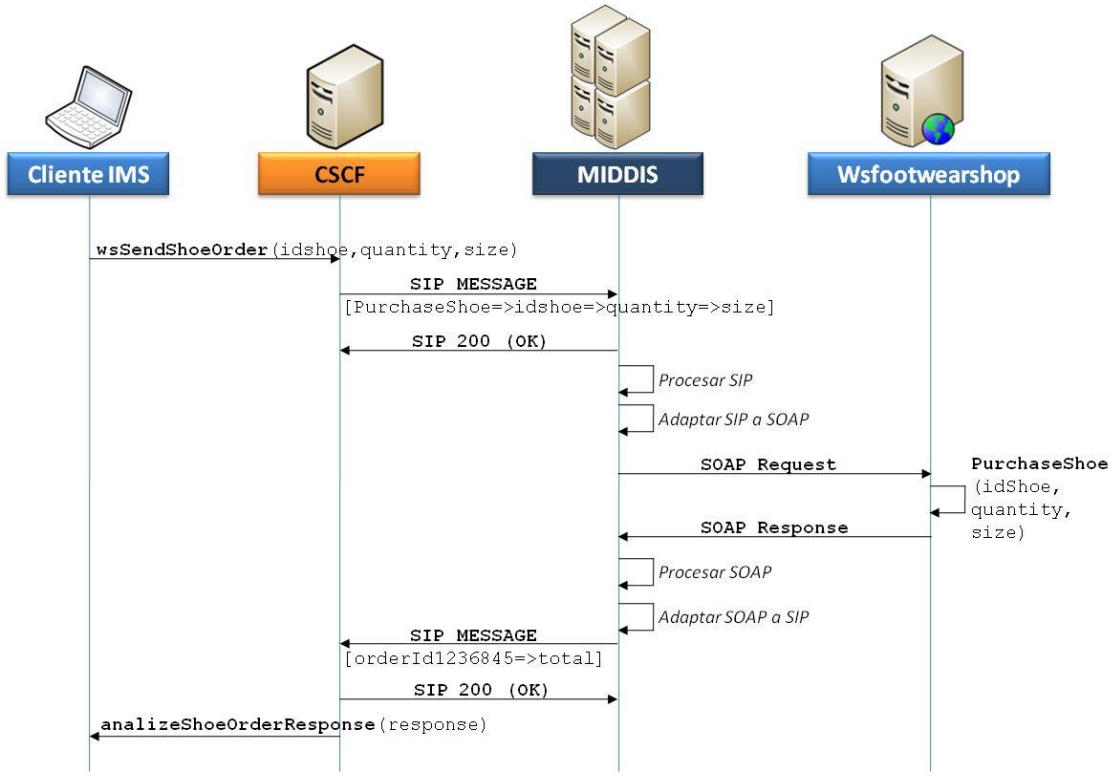


Figura 28. Invocación de un Servicio Web desde IMS

Para el caso de invocación de la funcionalidad de procesamiento de una orden de compra, el Servicio Convergente IMS/WS solicita al CSCF el envío de la misma a MIDDIS, donde dicha orden de compra está contenida en el cuerpo de un mensaje de petición SIP de tipo `MESSAGE` (con el nombre de la operación a invocar, y los parámetros que contiene la orden de compra). MIDDIS procesa esta petición SIP y procede de la siguiente manera: primero envía la respuesta SIP de tipo `200` (OK) al UAC SIP que generó la petición, luego obtiene los datos de invocación al WS del cuerpo de la petición, y por último adapta la petición SIP a una petición SOAP, que es enviada al `WSfootwearshop`. Entonces, el WS procesa la orden de compra, envía de regreso una respuesta SOAP con los datos del envío del artículo, y MIDDIS se encarga de adaptarla a una petición SIP de tipo `MESSAGE`, en cuyo cuerpo se incluye el contenido del mensaje SOAP de respuesta a la invocación del WS, que finalmente es entregada al Cliente IMS. Como último proceso, en el acceso a un WS desde un Servicio IMS, el Cliente IMS por una parte responde a la petición SIP de tipo `MESSAGE` con el mensaje `200` (OK) correspondiente, y por otra parte procesa la respuesta a la invocación del WS y muestra el resultado al usuario.

4.2.3. Señalización

Para demostrar la interacción de Servicios basados en IMS y SOA en el proceso de creación de servicios Convergentes IMS/SOA, se han registrado las tramas o mensajes del protocolo SIP, intercambiados entre el entorno de las Telecomunicaciones e Internet a través de MIDDIS para los procesos de:

- Registro de un WS en IMS
- Inicio de sesión entre un Servicio IMS y uno Web
- Invocación de un WS desde IMS
- Finalización de la sesión entre un Servicio IMS y uno Web

En la Figura 29 se muestra el escenario de referencia en el que se realizaron las pruebas de señalización. En él se especifica la configuración SIP y SOAP, a nivel de direcciones y puertos IP, de cada uno de los elementos que conforman tanto a MIDDIS como a los elementos de la red subyacente.

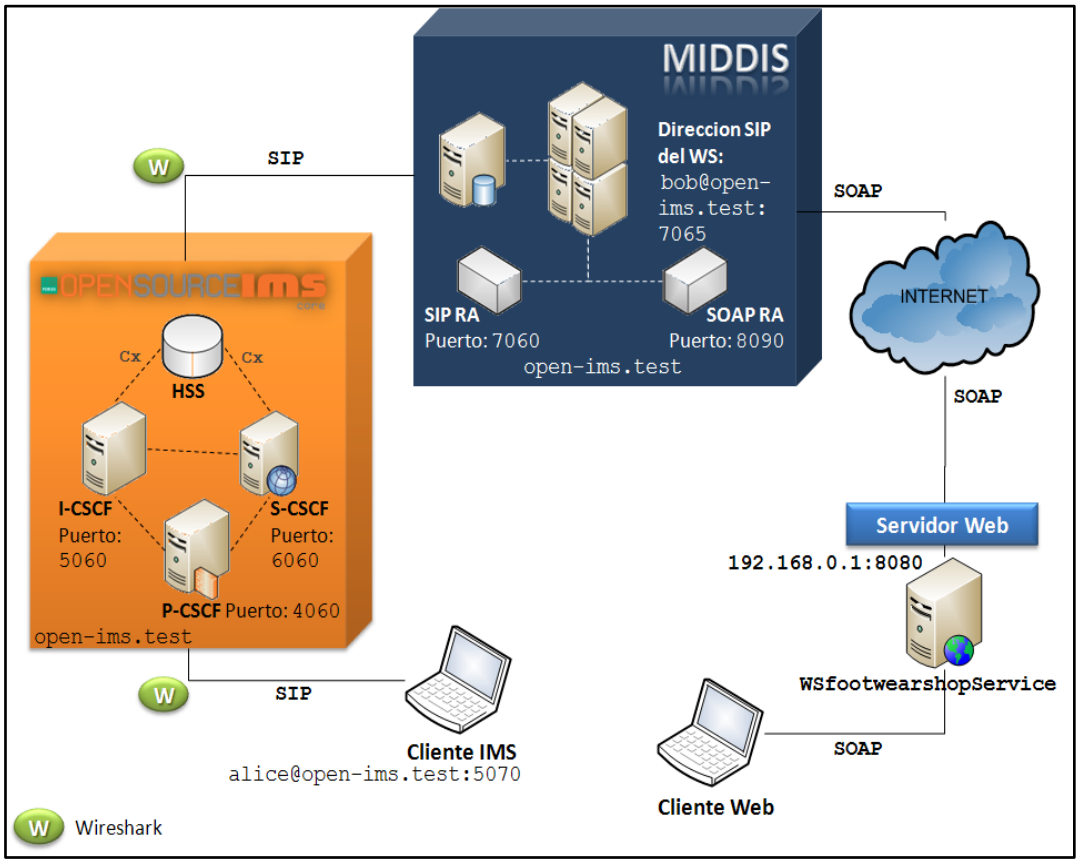


Figura 29. Escenario de Referencia en pruebas de señalización

El registro de datos se realizó utilizando la herramienta Wireshark versión 1.2.4. Este es un analizador de protocolos que permite ver todo el tráfico que pasa a través de la red a la cual se encuentra conectado el equipo en donde ha sido instalado. Esta herramienta es un software libre, desarrollado por Gerald Combs y otros expertos en redes (contribuidores).

Entre el OpenIMSCore y el Rhino SLEE SDK existen mensajes de señalización SIP. Dicho tráfico es visible en el trazado realizado por el Wireshark del equipo en donde se encuentran situados el Cliente IMS, el OpenIMSCore y el Rhino SLEE SDK.

- **Registro de un WS en IMS**

La información de registro de un Servicio Web en IMS está contenida en una petición SOAP con la siguiente estructura:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
<soa>
<operation>register</operation>
<serviceInfo>http://192.168.0.1:8080/ConvergedServices/WSfootwearsho
pService</serviceInfo>
</soa>
<ims>
<uaInfo>sip:bob@open-ims.test:7065</uaInfo>
</ims>
</Body>
</Envelope>
```

En la Figura 30 se muestran las tramas de señalización del proceso de Registro de un WS en IMS.

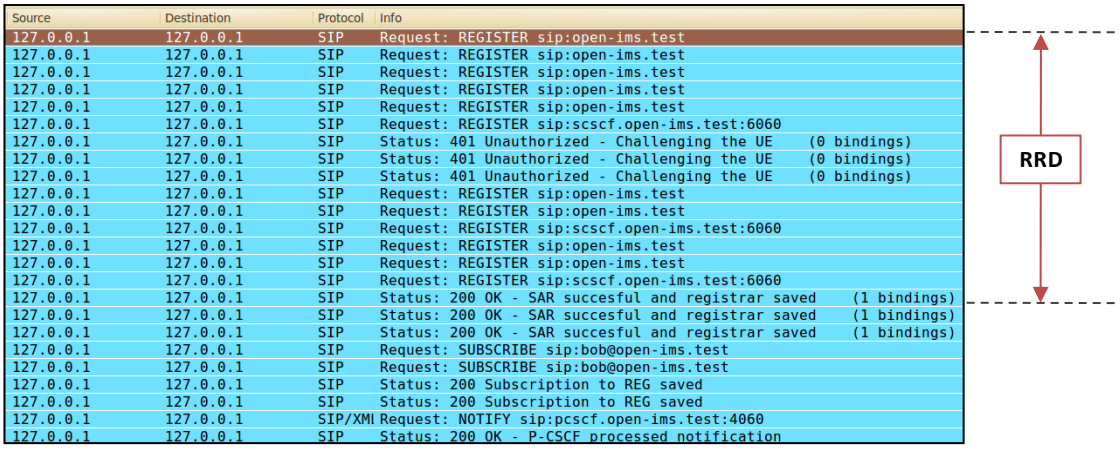


Figura 30. Tramas de señalización del proceso de Registro de un WS en IMS

En promedio, la medida del retardo en la respuesta del proceso de Registro de un WS en IMS a través de MIDDIS, o RRD, fue de 633 milisegundos. En la Figura 31 se muestra la gráfica obtenida del comportamiento de MIDDIS en el proceso de registro de un UA en IMS; dicho comportamiento es comparado con un escenario donde no es usado el mediador.

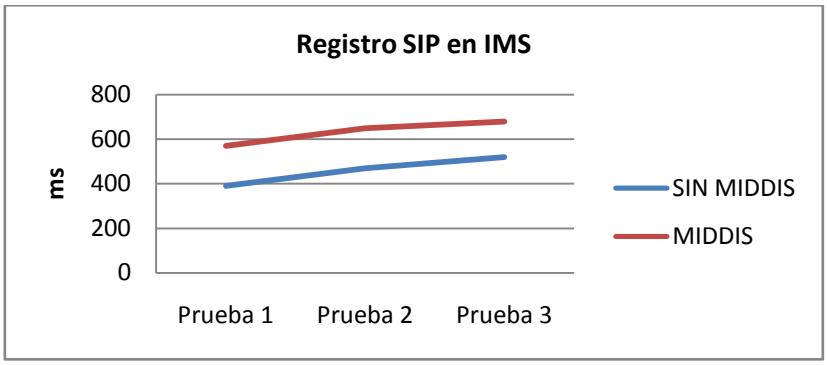


Figura 31. Gráfica del comportamiento obtenido con y sin MIDDIS en el proceso de registro de un UA SIP en IMS

Se puede observar que en este proceso el tiempo de respuesta de MIDDIS supera por 173 milisegundos, en promedio, al tiempo de respuesta obtenido en un ambiente sin mediador. Por lo tanto, teniendo en cuenta que un UAC que represente a un WS sólo se registrará una única vez en IMS, el Registro de un WS en IMS a través de MIDDIS se considera como un proceso de alto desempeño proveído por el mediador.

- **Inicio de sesión entre un Servicio IMS y uno Web**

Una vez se registran tanto el WS como el Cliente IMS en la red IMS, se puede proceder a realizar el inicio de sesión SIP entre los mismos. Para esto, desde el Cliente IMS se genera la petición SIP de tipo `INVITE` (Figura 32), haciendo clic en el botón "INVITE".

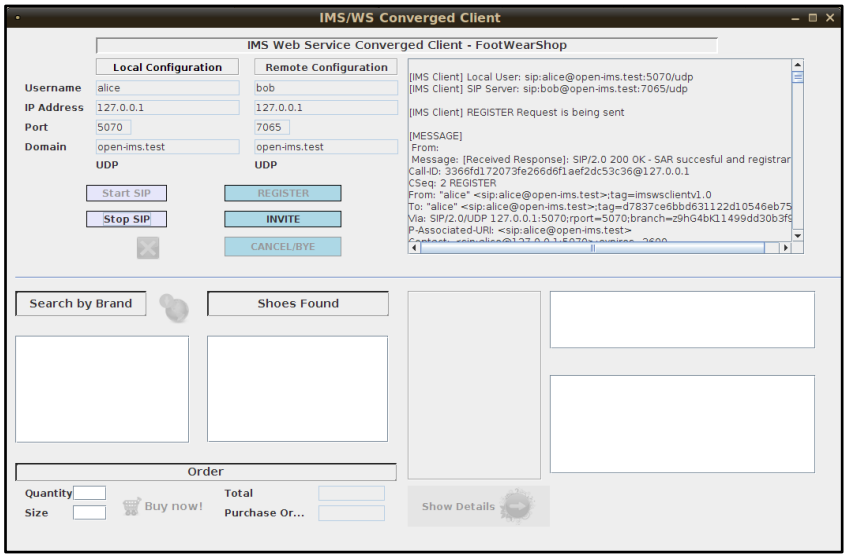


Figura 32. Interfaz de Usuario del Cliente IMS luego de registrarse en IMS

El resultado de esta ejecución es la habilitación, en el Cliente IMS, del acceso a las funcionalidades que provee el WS (Figura 33).

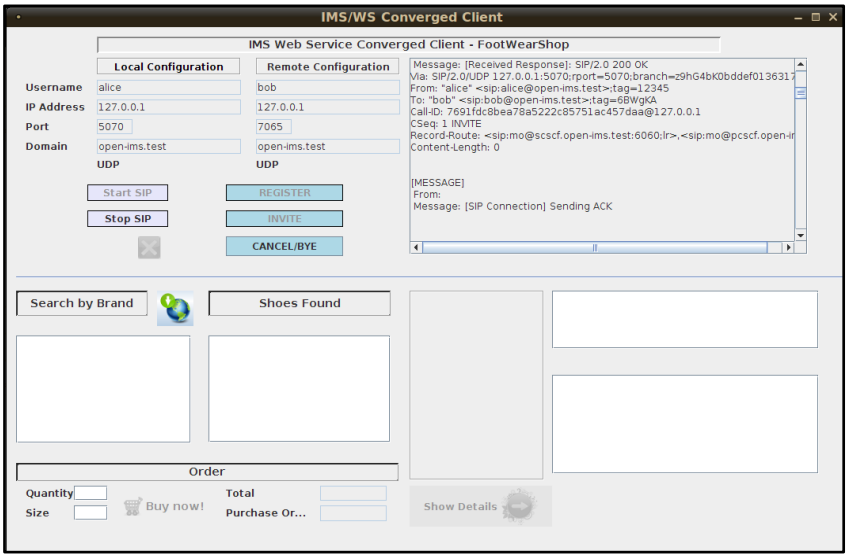


Figura 33. Interfaz de Usuario del Cliente IMS luego de iniciar sesión con el WS en IMS

A continuación, en la Figura 34, se muestran las tramas de señalización del proceso de Inicio de sesión entre un Servicio IMS y un Servicio Web, a través de MIDDIS.

Source	Destination	Protocol	Info
127.0.0.1	127.0.0.1	SIP	Request: INVITE sip:bob@127.0.0.1:7065
127.0.0.1	127.0.0.1	SIP	Status: 100 trying -- your call is important to us
127.0.0.1	127.0.0.1	SIP	Request: INVITE sip:bob@127.0.0.1:7065
127.0.0.1	127.0.0.1	SIP	Status: 100 trying -- your call is important to us
127.0.0.1	127.0.0.1	SIP	Request: INVITE sip:bob@127.0.0.1:7065
127.0.0.1	127.0.0.1	SIP	Request: INVITE sip:bob@127.0.0.1:7065
127.0.0.1	127.0.0.1	SIP	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	Status: 200 OK
127.0.0.1	127.0.0.1	SIP	Status: 200 OK
127.0.0.1	127.0.0.1	SIP	Status: 200 OK
127.0.0.1	127.0.0.1	SIP	Request: ACK sip:127.0.1.1:7060;transport=UDP;oc-node=101
127.0.0.1	127.0.0.1	SIP	Request: ACK sip:127.0.1.1:7060;transport=UDP;oc-node=101
127.0.0.1	127.0.1.1	SIP	Request: ACK sip:127.0.1.1:7060;transport=UDP;oc-node=101

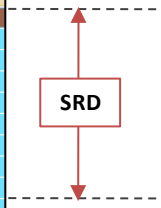


Figura 34. Tramas de señalización del proceso de Inicio de sesión entre un Servicio IMS y un Servicio Web

En promedio, el SRD o la medida del retardo en la respuesta del proceso Inicio de Sesión entre un Servicio IMS y uno Web, a través de MIDDIS, fue de 390 milisegundos. En la Figura 35 se muestra la gráfica obtenida del comportamiento de MIDDIS en el proceso de inicio de sesión entre un UA SIP (IMS) y un WS; dicho comportamiento es comparado con un escenario donde no es usado el mediador.

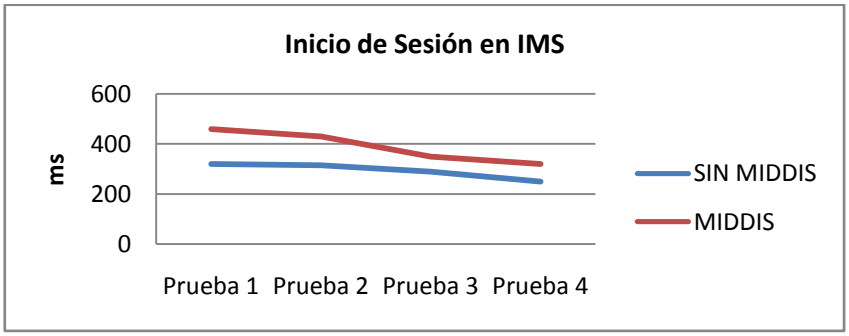


Figura 35. Gráfica del comportamiento obtenido con y sin MIDDIS en el proceso de inicio de sesión entre dos UA SIP

Se puede observar que en este proceso el tiempo de respuesta de MIDDIS supera por 97 milisegundos, en promedio, al tiempo de respuesta obtenido en un ambiente sin mediador. Teniendo en cuenta lo expuesto en (Joines, y otros, 2002) y (Malas, y otros, 2011), y considerando que el inicio de sesión entre dos UA SIP está enmarcado dentro de un escenario de convergencia entre los entornos Web y de Telecomunicaciones, se determinó que la propuesta del mediador en la ejecución de dicho proceso tiene un comportamiento bueno.

- **Invocación de un WS desde IMS**

El Cliente IMS puede invocar en el WS los diferentes métodos que le permiten obtener información y generar órdenes de compra de los productos, en este caso zapatos, que ofrece la tienda virtual implementada por medio del WS.

Al hacer clic en el botón de búsqueda de marcas disponibles se genera la invocación de una funcionalidad del WS desde el Cliente IMS, a través de la cual se obtienen las actuales marcas de zapatos disponibles en la tienda virtual, Figura 36. En las Figuras 37 y 38 se muestran los resultados de otras invocaciones.

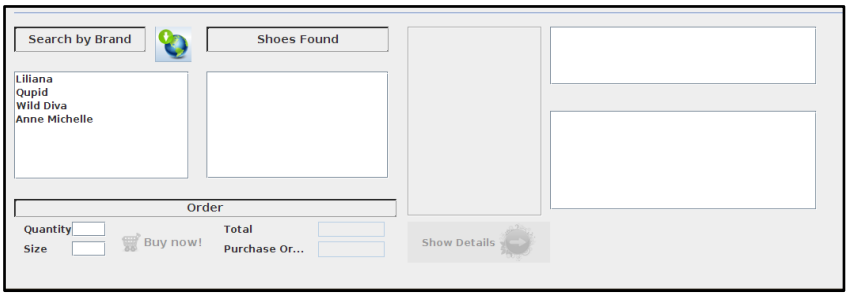


Figura 36. Invocación de un WS desde el Cliente IMS: Marcas disponibles de Zapatos

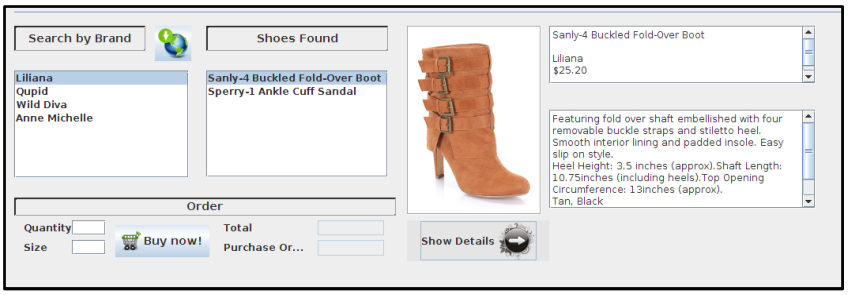


Figura 37. Invocación de un WS desde el Cliente IMS: Detalles de un Zapato

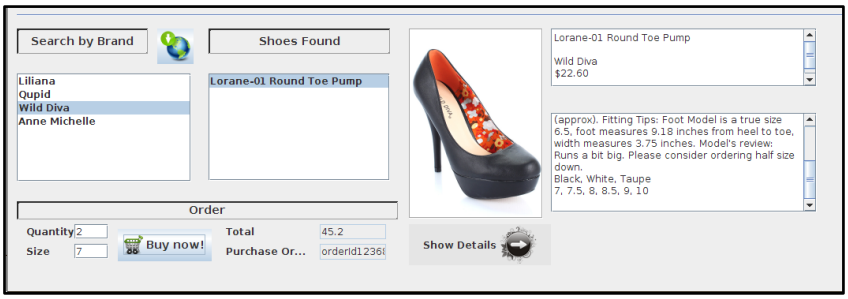


Figura 38. Invocación de un WS desde el Cliente IMS: Compra de un Zapato

En la Figura 39 se muestra el mensaje SIP de tipo `MESSAGE` a través del cual se envía la invocación a un método del WS.

```
Session Initiation Protocol
  Request-Line: MESSAGE sip:bob@open-ims.test:7065;transport=udp SIP/2.0
    Method: MESSAGE
    Request-URI: sip:bob@open-ims.test:7065;transport=udp
    [Resent Packet: False]
  Message Header
    Route: <sip:orig@scscf.open-ims.test:6060;lr>
    Call-ID: 3aef08e9b76ac9d085638def6413f124@127.0.0.1
    CSeq: 1 MESSAGE
    From: "alice" <sip:alice@127.0.0.1:5070>;tag=ClienteSIP
    To: "bob" <sip:bob@open-ims.test:7065>
    Via: SIP/2.0/UDP 127.0.0.1:4060;branch=z9hG4bK7e93.95ed9181.0
    Via: SIP/2.0/UDP 127.0.0.1:5070;rport=5070;branch=clientesip-3aef08e9b76ac9d085638def6413f124-127.0.0.1-1-me
    Max-Forwards: 16
    Contact: "alice" <sip:alice@127.0.0.1:5070>
    Content-Type: text/plain
    Content-Length: 23
    P-Asserted-Identity: <sip:alice@open-ims.test>
    P-Charging-Vector: icid-value="P-CSCFabcd000000004d4d827200000008";icid-generated-at=127.0.0.1;orig-ioi="ope
  Message Body
    Line-based text data: text/plain
      AvailableShoes=>Liliana
```

Figura 39. Petición SIP de tipo MESSAGE enviada por el Cliente IMS a MIDDIS con la invocación al WS

En la Figura 40 se muestra el mensaje SIP de tipo `MESSAGE` a través del cual se envía la respuesta a la invocación de un método del WS.

```
▷ Frame 146 (672 bytes on wire, 672 bytes captured)
▷ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▷ Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▷ User Datagram Protocol, Src Port: 7060 (7060), Dst Port: vtsas (5070)
▽ Session Initiation Protocol
  ▽ Request-Line: MESSAGE sip:alice@open-ims.test:5070;transport=udp SIP/2.0
      Method: MESSAGE
    ▷ Request-URI: sip:alice@open-ims.test:5070;transport=udp
      [Resent Packet: False]
  ▽ Message Header
    ▷ Via: SIP/2.0/UDP open-ims.test:7065;branch=z9hG4bKasqTotTYU-mRPJDk-okJYw;rport
    ▷ From: "bob" <sip:bob@127.0.0.1>;tag=rhinoimsclientv1.0
    ▷ To: "alice" <sip:alice@open-ims.test>
      Call-ID: RRhxPChUB37Zxs7PoBr_kw
    ▷ CSeq: 1 MESSAGE
      Max-Forwards: 70
      Content-Type: text/plain
      Content-Length: 237
    ▷ Contact: "bob" <sip:bob@127.0.0.1:7065>
  ▽ Message Body
    ▽ Line-based text data: text/plain
        0=>Sanly-4 Buckled Fold-Over Boot=>Liliana=>$25.20=>http://www.urbanog.com//images/products/2010/10a/sanly-4-tan_01.jpg;>5
```

Figura 40. Petición SIP de tipo MESSAGE enviada por MIDDIS al Cliente IMS con el resultado de la invocación al WS

En promedio, la duración del procesamiento y respuesta a la Invocación de un Servicio Web desde un Servicio IMS, a través de MIDDIS, está en el rango de 68 a 370 milisegundos. En la Figura 41 se muestra la gráfica obtenida del comportamiento de MIDDIS en el proceso de invocación de un WS desde un Servicio IMS. De acuerdo con lo expuesto en (Joines, y otros, 2002), y considerando que la invocación de un WS desde un Cliente IMS está enmarcada dentro de un escenario de convergencia entre los entornos Web y de Telecomunicaciones, se determinó que la propuesta del mediador en la ejecución de dicho proceso tiene un comportamiento bueno.

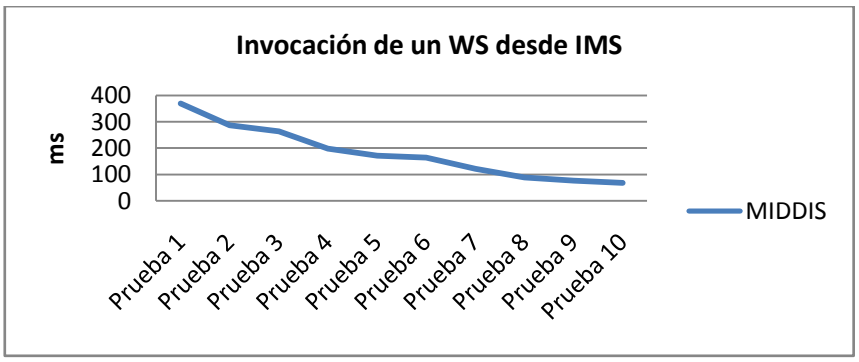


Figura 41. Gráfica del comportamiento de MIDDIS durante el proceso de invocación de un WS desde un Servicio IMS

- **Finalización de la sesión entre un Servicio IMS y uno Web**

El Cliente IMS genera la petición de fin de la sesión con el WS. En la Figura 42 se muestran las tramas de señalización del proceso de Finalización de sesión entre un Servicio IMS y un Servicio Web, a través de MIDDIS.

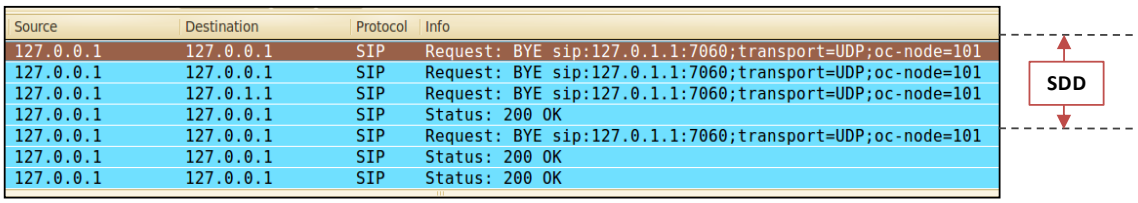


Figura 42. Tramas de señalización del proceso de Fin de sesión entre un Servicio IMS y un Servicio Web

En promedio, el SDD o la medida del retardo en la respuesta del proceso de Finalización de la Sesión entre un Servicio IMS y uno Web, a través de MIDDIS, fue de 129 milisegundos. En la Figura 43 se muestra la gráfica obtenida del comportamiento de MIDDIS en el proceso de finalización de sesión entre un UA SIP (IMS) y un WS; dicho comportamiento es comparado con un escenario donde no es usado el mediador. Se puede observar que en este proceso el tiempo de respuesta de MIDDIS supera por 29 milisegundos, en promedio, al tiempo de respuesta obtenido en un ambiente sin mediador. Por lo anterior, y teniendo en cuenta (Joines, y otros, 2002), se considera que mediador presenta un buen comportamiento durante este proceso.

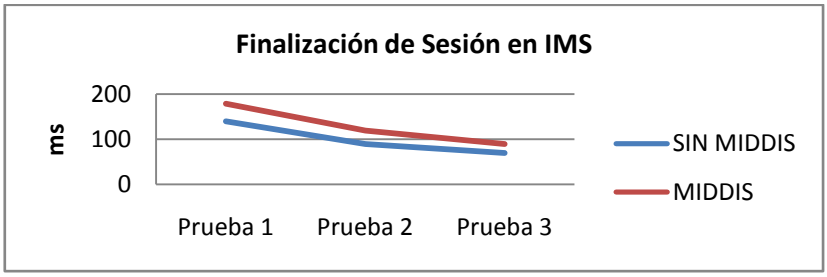


Figura 43. Gráfica del comportamiento obtenido con y sin MIDDIS en el proceso de finalización de sesión entre dos UA SIP

4.2.4. Publicación, consulta y modificación de las descripciones de los Servicios IMS y Web en MIDDIS

A continuación se presentan los procesos de publicación y consulta de las descripciones de los Servicios IMS y Web en el registro basado en UDDI de MIDDIS, ejecutados como peticiones SOAP desde el Cliente Web.

- **Publicación de las descripciones de los Servicios IMS y Web en MIDDIS**

La información de descripción de un Servicio (IMS o Web) para su publicación en MIDDIS está contenida en una petición SOAP con la siguiente estructura:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
<soa>
<operation>setUddi</operation><webServiceName>WSfootwearshopService</w
ebServiceName>
<uddiInfo>
      <businessEntityName>Entity Name</businessEntityName>
      <businessEntityDescription>Entity
Description</businessEntityDescription>
      <businessEntityContactPhone>Entity Contact
```

```
Phone</businessEntityContactPhone>
      <businessEntityContactEmail>Entity Contact
Email</businessEntityContactEmail>
      <businessServiceName>Service Name</businessServiceName>
      <businessServiceDescription>Service
Description</businessServiceDescription>
      <bindingTemplateLocation>Template
Location</bindingTemplateLocation>
      <bindingTemplateArchitecture>Template
Architecture</bindingTemplateArchitecture>
      <bindingTemplateTModel>Template TModel</bindingTemplateTModel>
</uddiInfo>
</soa>
</Body>
</Envelope>
```

- **Consulta de las descripciones de los Servicios IMS y Web en MIDDIS**

Para realizar la consulta de la información de descripción de un Servicio (IMS o Web) en MIDDIS se puede utilizar una petición SOAP donde se envíe vacío el parámetro de nombre del perfil del servicio, o donde se especifique el nombre del perfil del servicio, y de esta manera se obtendrán los nombres de los perfiles de los servicios registrados en MIDDIS y la descripción del servicio respectivamente:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
            <soa>
                  <operation>getUddi</operation>
                  <webServiceName>-</webServiceName>
                  <!-- Cuando se especifica el nombre del perfil -->
                  <webServiceName>WSfootwearshopService</webServiceName>
            </soa>
      </Body>
</Envelope>
```

Capítulo 5

CONCLUSIONES Y APORTES

5.1. Aportes

En resumen los principales aportes de esta tesis de maestría fueron:

- **Análisis detallado de las posibles soluciones existentes para la creación y despliegue de Servicios basados en SOA e IMS**. Los principales resultados de este análisis fueron: a) Un análisis y valoración de los antecedentes en la Convergencia de servicios IMS basados en SIP y Servicios Web (SOAP), provenientes tanto del mundo de las Telecomunicaciones como del mundo de Internet; b) Un análisis y valoración de los antecedentes para el Manejo de Sesiones en los Servicios Web, apuntando a soluciones que involucrarán tanto tecnologías de WS e Internet, como de las Telecomunicaciones; c) Un análisis y valoración de los mediadores para Telecomunicaciones, orientada a escoger una lógica de mediación con soporte tanto a servicios basados en IMS como a servicios basados en SOA.

- **Arquitectura de referencia para la mediación en la interacción de servicios basados en IMS y SOA**. Se propuso una arquitectura de referencia enfocada a proporcionar las funcionalidades de interacción, tanto a nivel de señalización y control como a nivel de acceso a servicios, entre SIP (del entorno de las Telecomunicaciones) y los WS (del entorno Internet), para la creación de Servicios Convergentes IMS/SOA. Por lo tanto, la arquitectura de referencia de la lógica de mediación provee un acceso a los servicios de cada entorno independiente de sus propios protocolos.

- **Implementación de Referencia**. Se realizó una instanciación de la Arquitectura de Referencia, diseñando una implementación de referencia cuyo alcance se acotó en el desarrollo de las funcionalidades básicas de mediación para la interacción de servicios Web e IMS. La evaluación inicial del mediador creado se llevó a cabo a través de un piloto de Servicio Convergente, diseñado y desarrollado para consumir las capacidades de un Servicio SOA sobre una red IP y las capacidades de un servicio de telecomunicaciones sobre una red IMS. Dicho prototipo permitió demostrar la eficiencia, eficacia y validez de la lógica de mediación propuesta para la interacción de Servicios basados en IMS y SOA.

- **Registro para la gestión de las descripciones de Servicios Web e IMS basada en UDDI.** Se realiza una propuesta inicial para la creación de un registro basado en UDDI que se encargue de la gestión centralizada de las descripciones de los WS, así como de las descripciones de los Servicios basados en IMS, y permita a los desarrolladores, de cualquiera de estos tipos de servicios, la publicación, modificación, y consulta de sus interfaces, para el desarrollo de Servicios Convergentes.

- **Servicios JSLEE para la Adaptación SIP/SOA**. Se proporcionan los servicios de adaptación SIP/SOAP, basados en JSLEE, para los procesos de: Registro de un WS en IMS; Inicio,

mantenimiento y finalización de sesión entre un Servicio IMS y uno Web; Acceso a las funcionalidades de un WS desde un Servicio IMS.

- **Modelamiento del comportamiento de los protocolos SIP y SOAP a través de VFSM**. Se define y ejecuta el comportamiento de los protocolos de comunicaciones SIP y SOAP a través de Máquinas Virtuales de Estados Finitos. La importancia del uso de FSM es que éstas se pueden extender a través de FSM más complejas, manteniendo la independencia entre el modelo y el código de implementación.

5.2. Conclusiones

MIDDIS permite la creación y despliegue de Servicios Convergentes IMS/SOA, asegurando la participación de los operadores de Telecomunicaciones y de Internet en el control de la lógica de los servicios, así como también facilita y agiliza su implementación a los desarrolladores de aplicaciones, para ambos entornos, y donde el común denominador, en el actual paradigma de desarrollo de servicios centrados en el usuario, es la rápida provisión de aplicaciones cada vez más complejas. Con esto se busca, principalmente, mantener la relevancia de los operadores de Telecomunicaciones en la cadena de valor de los futuros Servicios Convergentes, pues es un contexto en el que ha perdido espacio, debido a la complejidad en el desarrollo de los servicios de Telecomunicaciones, permitiendo que Internet tome su lugar donde ya no se considera indispensable y/o donde puede ser reemplazado con tecnologías estandarizadas, mucho más fáciles de aprender y de utilizar.

Cada vez más se están proponiendo y desarrollando escenarios para la implementación de servicios en entornos pre-IMS/IMS, lo que permite demostrar que esta convergencia es cada vez más factible y necesaria. MIDDIS aborda la convergencia de servicios a nivel de interacción bajo un enfoque arquitectónico que incluye aspectos tanto de SOA como de IMS. En este sentido, se basa en la naturaleza versátil de SIP y de SOAP, extendiéndolos y complementándolos, para integrar a IMS con SOA en su capa de aplicaciones, y con ello proporcionar una gran variedad de oportunidades en la creación de servicios convergentes. De esta manera se proporciona un medio adecuado para la prestación de los servicios necesarios para justificar las inversiones en IMS.

A diferencia de la primitiva INFO (Donovan, 2000), otro mecanismo de SIP para el intercambio de mensajes de señalización durante la llamada, la petición `MESSAGE` puede transportar varios tipos de contenidos (basándose en la codificación MIME) entre un conjunto de participantes, casi en tiempo real, y esto lo puede realizar tanto dentro como fuera de una sesión SIP. Debido a sus características y capacidades de usabilidad, en MIDDIS se optó por la utilización del tipo de petición `MESSAGE` de SIP para la invocación de los métodos que proporciona un WS, desde un Servicio IMS.

Los WS no se identifican, descubren y localizan por medio de direcciones SIP, sin embargo la interacción entre servicios basados en IMS y SOA se puede realizar mediante la iniciación de sesiones SIP, a través de MIDDIS.

Es posible gestionar las descripciones de los servicios IMS y Web de manera unificada, a través del modelo de UDDI de los WS, para proporcionar un medio centralizado que pueda ser

utilizado por los desarrolladores de ambos tipos de servicios en el desarrollo de las futuras generaciones de Servicios Convergentes.

El piloto de Servicio Convergente desarrollado permitió evaluar la funcionalidad de MIDDIS, validar el uso del protocolo SIP-IMS y SOAP, y obtener un sistema más robusto al llevar a cabo la retroalimentación de la Lógica de Mediación, con base en los resultados obtenidos en las pruebas.

La propuesta e implementación de una la Lógica de Mediación para la Interacción de Servicios basados en SOA e IMS se basa en tecnologías abiertas y estandarizadas, que gracias a sus características de extensibilidad permiten proveer una simple, nueva, y muy buena solución a los problemas de interconexión y de convergencia de redes y servicios que se presentan actualmente, y sin afectar su carácter abierto y estandarizado.

A partir de la investigación y el análisis de las propuestas más importantes alrededor de la interacción entre IMS (SIP) y los Servicios Web, surge MIDDIS como un fundamento promisorio para la convergencia de Servicios IMS y Web, bajo un entorno de Red de Próxima Generación.

5.3. Trabajos Futuros

Esta tesis de maestría ha aportado soluciones al problema de la Interacción de Servicios basados en IMS y SOA, proporcionando una Lógica de Mediación que realiza la adaptación de la capa de señalización y control, y de la capa de datos de acceso a los servicios, del entorno IMS y Web, para permitir la creación y despliegue de Servicios Convergentes IMS/SOA. Por lo tanto, teniendo en cuenta el campo de estudio de este proyecto de grado se propone los siguientes trabajos futuros:

- **Implementación del Subsistema de Medios**

Con el fin de proporciona una Lógica de Mediación que soporte la adaptación del acceso a cualquier tipo de servicio, tanto IMS como Web, desde cualquier entorno, de Telecomunicaciones o SOA, se debe implementar la funcionalidad propuesta de este subsistema. En este sentido, las Funciones de Recursos Multimedia de IMS, y los Adaptadores de Recursos de Medios para JAIN SLEE, son dos campos tecnológicos que deben ser abordados.

- **Implementación de la especificación WS-Context**

Con el fin de proporcionar una gestión real del contexto de la sesión de un WS, para su uso efectivo no sólo en el entorno SOA sino también en el entorno de las Telecomunicaciones, a través de MIDDIS, se debe realizar la implementación de esta especificación en el entorno SOA, y la extensión de MIDDIS para los procesos de gestión requeridos desde el entorno de las Telecomunicaciones.

- **Extensión de SIP y SDP para el manejo de contenido application/soap+xml**

En el presente trabajo de maestría se implementó el protocolo SIP básico, con sus extensiones estándares para IMS. No obstante se considera que la invocación de los WS desde IMS, por medio de MIDDIS, puede ser facilitada a través de la generación de comandos de invocación a

servicios basados directamente en SOAP, y transportados en el tipo de petición `MESSAGE` de SIP, utilizando el encabezado de tipo de contenido: `Content-Type: application/soap+xml`. Así mismo, se necesitará extender a SDP para la definición de los medios que pueden ser intercambiados entre ambos entornos, y para otras tareas de descripción de la sesión que mejoren y faciliten la usabilidad del mecanismo de MIDDIS.

- **Extender el modelo de gestión de las descripciones de los Servicios IMS y Web basado en UDDI de MIDDIS**

Se considera que la centralización de los procesos de publicación, modificación y consulta de las interfaces de un servicio IMS o Web, basada en el modelo UDDI de los WS, es fundamental para su uso desde cualquier entorno, por parte de cualquier tipo de desarrollador, y con el fin de que éstos últimos creen rápida y fácilmente un sinnúmero de Servicios Convergentes IMS/SOA.

Referencias

AePONA. 2005. VAS Implementation in the IMS. [En línea] 2005. [Citado el: 5 de Mayo de 2007.] http://www.mcubedigital.com/secure/download_resource.aspx?ID=4.

Al-Begain, K., Balakrishna, C., Moro, D., Galindo, L. A. 2009. IMS: A Development and Deployment Perspective. Innovative services and business models through WIMS 2.0 initiative. s.l. : John Wiley & Sons, ISBN: 978-0-470-74034-7, Octubre de 2009.

Almeida, Y. 2007. Plataforma para el establecimiento y desarrollo de conferencias multimedia. [En línea] 2007. [Citado el: 8 de Mayo de 2009.] http://www.cujae.edu.cu/Eventos/CITTEL/Memorias/CITEL2004/Trabajos/CIT007.pdf.

Andrinal, J. M., Martínez, J. F., García, A. B. 2007. Arquitecturas Orientadas a Servicios en Redes de Nueva Generación. [En línea] 2007. [Citado el: 16 de Abril de 2008.] http://www.di.unc.edu.ar/collecter2007/papers/19.pdf.

Baravaglio, A., Licciardi, C. A., Venezia, C. 2005. Web Service Applicability in Telecommunication Service Platforms. International Journal of Web Services Practices, Vol.1, No.1-2. 2005, pp. 167-172.

Basicevic, L., Popovic, M., Velikic, I. 2010. Use of Finite State Machine Based Framework in Implementation of Communication Protocols – A Case Study. IEEE Computer Society, 2010 Sixth Advanced International Conference on Telecommunications (AICT), 2010, pp. 161-166. 2010.

Box, D., y otros. 2004. Web Services Addressing (WS-Addressing). [En línea] W3C, 10 de Agosto de 2004. [Citado en: 21 de Mayo de 2007.] http://www.w3.org/Submission/ws-addressing/.

Camarillo, G. y García, M. A. 2006. The 3G IP Multimedia Subsystem (IMS) Merging the Internet and the Cellular Worlds. s.l. : John Wiley & Sons, ISBN-13 978-0-470-01818-7, 2006.

Carnegie Mellon. 2007. Middleware. [En línea] Carnegie Mellon, 2007. [Citado el: 21 de Mayo de 2007.] http://www.sei.cmu.edu/str/descriptions/middleware.html.

Carrascosa, A. 2005. Nuevos Servicios Multimedia, Convergencia y Evolución con IMS. [En línea] Agosto, Septiembre de 2005. [Citado el: 10 de Octubre de 2007.] http://www.coit.es/publicaciones/bit/bit152/84-87.pdf.

Carvalho de Gouveia, F. y Magedanz, T. 2008. Impacts of LTE on emerging NGN Service Architectures. [En línea] Technische Universität Berlin, Fraunhofer Institut FOKUS, Mayo de 2008. [Citado el: 29 de Agosto de 2008.] http://www.comnets.uni-bremen.de/itg/itgfg521/aktuelles/fg-workshop-08052008/Gouveia_TU_Berlin_IMS.pdf.

Cheng, B., Guo, J., Meng, X., Chen, J. 2008. SIP Based Real-Time Web Services Communication Model. Computing, Communication, Control, and Management, 2008.CCCM '08.ISECS International Colloquium on. 29 de Agosto de 2008, págs. 439 - 443.

Chou, W., Li, L., Feng, L. 2006. WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP. International Conference on Web Services (ICWS '06). Septiembre de 2006, págs.515-522.

Chou, W., Liu, F., Li, L. 2010. Web Service for Tele-Communication. IEEE Computer Society, Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006). 2006.

Costello, R. L. 2002. REST (Representational State Transfer). [En línea] 2002. [Citado el: 15 de Enero de 2009.] http://www.xfront.com/sld001.htm.

Cruz, A. 2005. Una nueva convergencia: ¿Java en la red?. [En línea] InfoWorld, 3 de Agosto de 2005. [Citado el: 14 de Enero de 2009.] http://www.iworld.com.mx/iw_SpecialReport_read.asp?iwid=3827&back=2&HistoryParam=U.

Curbera, F., Nagy W., Weerawarana S. 2001. WebServices: Why and How. Proc. Wksp. Object Orientation and WebServices OOWS2001, Agosto 9 de 2001.

Deason, N. 2001. White Paper SIP and SOAP, White Paper Release 1.0. [En línea] Ubiquity, Mayo de 2001. [Citado el: 7 de Marzo de 2008.] http://www.voip-info.de/res/Technologie/Ubiquity_SIP_and_SOAP.pdf.

Dong, W., y Newmarch, J. 2005. Adding Session and Transaction Management to Web Services by using SIP. [En línea] 2005. [Citado el: 7 de Marzo de 2008.] http://jan.newmarch.name/publications/sip-soap.pdf.

Donovan, S. 2000. The SIP INFO Method. [En línea] IETF, Octubre de 2000. [Citado el: 15 de Mayo de 2009.] http://tools.ietf.org/html/rfc2976.

Dornan, A. 2007. Tech Road Map: Oasis Takes On Web Services Session Management. [En línea] InformationWeek, 8 de Septiembre del 2007. [Citado el: 8 de Mayo de 2009.] http://www.informationweek.com/news/software/soa/showArticle.jhtml?articleID=201804545.

Expocomm Argentina. 2005. Seminario de Tecnología y Mercado. [En línea] 2005. [Citado el: 14 de Abril de 2005.] http://www.ejkreed.com/extras/img/web_727_brochurestm.pdf.

Falcarin, P. y Venezia, C. 2008. Communication Web services and JaiN-slee integration Challenges. International Journal of Web Services Research. Octubre-Diciembre, 2008, pp. 59-78.

Fernández, J. A. 2007. SOA y la experiencia de BEA en otros sectores. [En línea] BEA Systems Iberia, 2007. [Citado el: 16 de Abril de 2008.] http://www.financialtech-mag.com/_docum/iBanca_Presentacion__BEA.pdf.

Fornies, H. B., Garcia, A. M., Laliena, G. C., Sanclemente, L. 2007. System for invoking Web Services by means of SIP signaling. [En línea] United States Patent Application Publication, 6 de Septiembre de 2007. [Citado el: 29 de Agosto de 2008.] http://www.freepatentsonline.com/US20070208804.pdf.

Fraunhofer FOKUS. 2009. Open IMS Core. [En línea] Fraunhofer FOKUS, 2009. [Citado en: Enero de 2009.] http://www.openimscore.org/.

Gehlen, G., Aijaz, F., Zhu, Y., Walke, B. 2006. Mobile P2P Web Service Creation using SIP. [En línea] RWTH Aachen University, Faculty 6, Communication Networks, Kopernikusstr, 2006. [Citado el: 1 de Octubre de 2007.] http://www.comnets.rwth-aachen.de/typo3conf/ext/cn_download/pi1/passdownload.php?downloaddata=932%7C1.

Georgescu, S. 2007. IMS: An Architecture for Convergent Next Generation Multimedia Services. [En línea] Ericsson, 2007. [Citado el: 2 de Abril de 2008.] http://www.iaria.org/conferences2007/filesICSNC07/IMSConvergentMultimediaServices.ppt.

Gonzalez, J., Iglesias, C. A. y Echanique, F. 2009. La Plataforma Telcoblocks de Despliegue y Desarrollo de Servicios VoIP. [En línea] Universidad Politécnica de Madrid, Departamento Ingeniería de Sistemas Telemáticos, División de I+D+i Germinus XXI. [Citado el: 5 de Enero de 2010.] http://www.jonbaraq.eu/wordpress/wp-content/uploads/2009/07/telcoblocksjitel091.pdf.

Handley, M., y otros. 1999. SIP: Session Initiation Protocol. [En línea] IETF, Marzo de 1999. [Citado en: Abril de 2005.] http://www.ietf.org/rfc/rfc2543.txt.

IMS Laboratory, LaBRI, ENSEIRB. 2006. HomeSIP: building a smart Space with SIP Protocol. [En línea] IMS Laboratory: Laboratory of Microelectronics, Phoenix INRIA Bordeaux, School of Electrical Engineering, Computer Science and Telecommunications, Enero de 2006. [Citado el: 8 de Mayo de 2009.] http://www.enseirb.fr/cosynux/HomeSIP/.

Ivanov, I. 2006. Mobicents: JSLEE for the People, by the People. [En línea] java.net, 14 de Marzo de 2006. [Citado el: 30 de Enero de 2009.] http://today.java.net/pub/a/today/2006/03/09/mobicents-jslee.html.

Jähnert, J., Cuevas, A., Moreno, J. I., Villagrá, V. A., Wesner, S., Olmedo, V., Einsiedler, H. 2007. The Akogrimo way towards an extended IMS architecture.International Conference on Intelligence in Networks (ICIN2007). Emerging Web and Telecom Services: Collition or Competition. Bordeaux France 8-11 de Octubre de 2007.

JAINSLEE.org. 2005a. JAIN SLEE Fundamentals. [En línea] JAINSLEE.org, 2005. [Citado el: 30 de Enero de 2009.] http://www.jainslee.org/slee/fundamentals.html.

JAINSLEE.org. 2005b. JAIN SLEE and SIP Servlet.Comparison. [En línea] JAINSLEE.org, 2005. [Citado el: 30 de Enero de 2009.] http://jainslee.org/othertechnologies/sleevssipservlet.html.

Joines, S., Willenborg, R. y Hygh, K. 2002. Performance Analysis for Java Websites. s.l. : Addison Wesley, ISBN-13: 978-0201844542, 2002.

Kmatveev. 2008. Java middleware for telecom: JSLEE vs. SIP Servlets. [En línea] Blog en WordPress, 3 de Marzo de 2008. [Citado el: 18 de Enero de 2009.] http://technfun.wordpress.com/2008/03/03/java-middleware-for-telecom-jslee-vs-sip-servlets/.

Levenshteyn, R. y Fikouras, I. 2006. Mobile Services Interworking for IMS and XML WebServices. IEEE Communications Magazine (2006 IEEE). Septiembre de 2006, págs.80-87.

Light Reading. 2006. Service Orchestration, Service Orchestration in Use. [En línea] 2006. [Citado el: 9 de Abril de 2008.] http://www.lightreading.com/document.asp?doc_id=96054&page_number=8.

Lim, S., Ferry, D., O'Doherty, P., y Page, D. 2003. JAIN JAIN™ SLEE Tutorial. [En línea] Sun Microsystems, Open Cloud, 2003. [Citado el: 6 de Agosto de 2009.] http://java.sun.com/products/jain/JAIN-SLEE-Tutorial.pdf.

Little, M. 2005. A session & context concept for Web Services. [En línea] java.net, 7 de Mayo del 2005. [Citado el: 15 de Mayo de 2009.] http://weblogs.java.net/blog/marklittle/archive/2005/05/a_session_conte.html.

Liu, F., Chou, W., Li, L., Li, J. 2004. WSIP – Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP. [En línea] Avaya Labs Research, 2004. [Citado el: 29 de Agosto de 2008.] http://dpnm.postech.ac.kr/mcs/papers/ALR-2004-004-paper.pdf.

Liu, J., Jiang, S., Lin, H. 2006. Introduction to Diameter. [En línea] 2006. [Citado en: Abril de 2006.] http://www.ibm.com/developerworks/wireless/library/wi-diameter/.

Loos, C., Wesner, S., Jähnert, J. 2005. Specific Challenges of Mobile Dynamic Virtual Organizations. [En línea] Innovation and the Knowledge Economy: Issues, Applications, Case Studies (IOS Press, ISBN 1-58603-563-0, editors P. Cunningham, M. Cunningham), 2005. [Citado el: 23 de Febrero de 2010.] http://www.mobilegrids.org/modules2dd1.pdf?name=UpDownload&req=getit&lid=125.

Lucent Technologies. 2005. IP Multimedia Subsystem (IMS) Service Architecture. [En línea] 2005. [Citado en: Septiembre de 2005.] http://www.lucent.com/livelink/090094038005df2f_White_paper.pdf.

Magedanz, T. 2007. SOA in Telecoms – A dejà vu or a new research field?. [En línea] Teschnische Universität Berlin, Fraunhofer Institute FOKUS, Germany, 2007. [Citado el: 15 de Enero de 2009.] http://research.ihost.com/mw07/MNCNA-California-Invited%20Talk-26-11-2007.pdf.

Magedanz, T., Weik, P., Vingarzan, D., Carvalho de Gouveia, F., Wahle, S. 2008. Experiences on the Establishment and Provisioning of NGN/IMS Testbeds - The FOKUS Open IMS Playground and the Related Open Source IMS Core. [En línea] Fraunhofer FOKUS, Technical University of Berlin, Germany, 2008. [Citado el: 14 de Enero de 2009.] http://www.icin.biz/files/programmes/Session2B-2.pdf.

Malas, D., CableLabs, Morton, A., AT&T Labs. 2011. Basic Telephony SIP End-to-End Performance Metrics [En línea] IETF, Enero de 2011. [Citado en: Febrero de 2011.] http://www.faqs.org/rfcs/rfc6076.html.

Mansutti, D., Lamberton, M., Mathieu, M., Menez, B. 2007. Study Case on the convergence of Web Services and IMS Using Instant Communication Services. [En línea] OpenCall Software, Hewlett-Packard. [Citado el: 14 de Enero de 2009.] http://www.icin.biz/files/programmes/Session3B-1.pdf.

Maretzke, M. 2005. JAIN SLEE Technology Overview. [En línea] Maretzke.De, 12 de Abril de 2005. [Citado el: 30 de Enero de 2009.] http://www.maretzke.de/pub/lectures/JSLEE_Overview_2005/JSLEE_Overview_2005.pdf.

Maretzke, M. 2008. Java Telecommunication Application Server Technology Comparison. [En línea] 29 de Julio de 2008. [Citado el: 20 de Agosto de 2009.] http://www.maretzke.de/pub/whitepapers/telcoappserver_2008/Positioning_TelcoApplicationServer_Technologies_MiMa_v1.0_20080729.pdf.

Mariotto, F. T. 2006. Tendencias en Convergencia Móvil. [En línea] 2006. [Citado en: Abril de 2006.] http://www.comunidadmovil.com.co/media/Flavio%20Mariotto2.pdf.

McHugh, M. 2006. IMS & SOA - Driving the Future of Telecommunications. [En línea] 2006. [Citado el: 28 de Agosto de 2008.] http://www.tmcnet.com/ims/1206/industry-perspective-ims-and-soa-driving-the-future.htm.

Mikhanov, S. 2008. JAIN SLEE vs. SIP Servlets: Asynchronicity is not the cure. [En línea] 24 de Abril de 2008. [Citado el: 18 de Enero de 2009.] http://www.mikhanov.com/2008/04/24/jain-slee-vs-sip-servlets-asynchronicity-is-not-the-cure-26.

Moreno, M. M., Álvarez, C., Fernández, C. M., Vinyes, S. J. 2005. Propuesta de utilización de SIP como protocolo de señalización en la red de acceso radio de sistemas UMTS. [En línea] 2005. [Citado en: Abril de 2006.] http://www.ahciet.net/comun/portales/1000/10002/10007/10299/docs/011-72-80.pdf.

Moro, D., Lozano, D., y Galindo, L. A. 2008. WIMS 2.0: la convergencia del mundo Telco con la web 2.0. [En línea] 3 de Diciembre de 2008. [Citado el: 14 de Enero de 2009.] http://sociedadinformacion.fundacion.telefonica.com/DYC/SHI/seccion=1188&idioma=es_ES&id=2009100116310161&activo=4.do?elem=7518.

Nolle, T. 2008. Building revenue-increasing telecom services for the future. [En línea] SearchTelecom.com, 2008. [Citado el: 18 de Abril de 2009.] http://searchtelecom.techtarget.com/tip/0,289483,sid103_gci1320561_mem1,00.html.

Oasis. 2002. UDDI Version 2.03 Data Structure Reference. [En línea] Oasis, 19 de Julio de 2002. [Citado en: Agosto de 2010.] http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm#_Toc25130802.

Oasis. 2007. Web Services Context Specification (WS-Context) Version 1.0. [En línea] Oasis, 2 de Abril de 2007. [Citado en: 14 de Mayo de 2009.] http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html.

Oasis. 2009a. Web Services Coordination (WS-Coordination). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wscoor/2006/06.

Oasis. 2009b. Web Services Atomic Transaction (WS-AtomicTransaction). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wsat/2006/06.

Oasis. 2009c. Web Services Business Activity (WS-BusinessActivity). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wsba/2006/06.

Oasis. 2009d. OASIS Web Services Transaction (WS-TX) TC. [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 22 de Mayo de 2007.] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx.

Oh, S., Aktas, M. S., Pierce, M., y Fox, G. 2006. Optimizing Web Service Messaging Performance Using a Context Store for Static Data. Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics, Istanbul, Turkey. Mayo 27-29, 2006, pp. 334-342.

OpenCloud. 2009a. RHINO 2.1: OVERVIEW AND CONCEPTS. [En línea] OpenCloud, 30 de Marzo de 2009. [Citado el: 2 de Mayo de 2009.] https://developer.opencloud.com/devportal/download/attachments/40730782/OverviewAndConcepts.pdf?version=1.

OpenCloud. 2009b. FSM Tool Developer's Guide. [En línea] OpenCloud, 5 de Mayo del 2009. [Citado el: 6 de Mayo de 2009.] https://developer.opencloud.com/devportal/display/FSMD/FSM+Tool+Developer%27s+Guide.

OpenCloud. 2010. Rhino 2.1-03 release. [En línea] OpenCloud, 2010. [Citado en: Abril de 2010.] https://developer.opencloud.com/devportal/display/OCDEV/Rhino+2.1.

Parlay. 2009. Parlay. [En línea] 2009. [Citado en: Abril de 2009.] http://www.parlay.org.

Parlay X. 2009. Parlay X. [En línea] Ericsson, 2009. [Citado en: Abril de 2009.] http://www.ericsson.com/developer/sub/open/technologies/parlayx/about/parlayX.

PC-NEWS. 2006. El usuario final es quien decide el rumbo de la tecnología, Lucent Technologies afina sus estrategias en esa dirección. [En línea] 2006. [Citado el: 10 de Agosto de 2006.] http://www.pc-news.com/imprimir.asp?ida=2390.

Pechuán, L. M. 2004. El nuevo sistema multimedia conocido como IMS que adoptarán las redes UMTS. [En línea] Trabajo presentado en la asignatura Redes de Ordenadores, Curso 2004-2005, Universidad de Valencia. [Citado en: Octubre de 2005.] http://www.uv.es/~montanan/redes/trabajos/IMS.doc.

RADVISION. 2006. IMS SIP and Signaling, The RADVISION Perspective. [En línea] 2006. [Citado en: Agosto de 2006.] http://www.radvision.com/NR/rdonlyres/FC60D840-1FE5-4F82-A6A2-088D2D4AADCB/0/IMSSIPWhitePaper.pdf.

Rapeeporn, J. y Benchaphon, L. 2007. Reliable Session Management for Web Services by Using SIP.International Joint Conference on Computer Science & Software Engineering (JCSSE2007).Mayo 2-4 de 2007, págs.161-166.

Rezabeigi, K., Vafaei, A., Movahhedinia, N. 2008. A Web Services based Architecture for NGN Services Delivery. [En línea] World Academy of Science, Engineering and Technology, 2008. [Citado el: 14 de Octubre de 2009.] http://www.waset.org/journals/waset/v43/v43-86.pdf.

Rolan, C. y Hu, H. 2008. Defining a Service Delivery Platform Architecture using Generic IMS and SOA concepts. [En línea] Centre for Telecommunications Access and Services, School of Electrical and Information Engineering, University of the Witwatersrand, 2008. [Citado el: 18 de Abril de 2009.] http://www.icin.biz/files/programmes/Poster-1.pdf.

Rosenberg, J., y otros. 2002. SIP: Session Initiation Protocol. [En línea] IETF, Junio de 2002. [Citado en: Abril de 2005.] http://www.rfc-editor.org/rfc/rfc3261.txt.

Schwartz, S. 2006. IMS Needs Orchestration on Many Levels. [En línea] 1 de Julio de 2006. [Citado el: 9 de Abril de 2008.] http://www.billingworld.com/articles/feature/IMS-Needs-Orchestration-on-Many-Levels.html.

Serrano, C. 2005. Modelo Integral para el Profesional en Ingeniería. Popayán: Editorial Universidad del Cauca, 2005.

Sun Microsystems y OpenCloud. 2009. JAIN SLEE (JSLEE) 1.1 Specification, Final Release. [En línea] 2009. [Citado el: 2 de Mayo de 2009.] http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_JCP-Site/en_US/-/USD/VerifyItem-Start/jslee-1_1-fr-spec.pdf?BundledLineItemUUID=1VlIBe.lNGIAAAEh9Ds1wgvt&OrderID=EO1IBe.l0T0AAAEh4js1wgvt&ProductID=qsJIBe.pUsUAAAEbNVMkexRH&FileName=/jslee-1_1-fr-spec.pdf.

Tarkoma, S., Rovira, J., Postmann, E., Rajasekaran, H., Kovacs, E. 2008. Creating Converged Services for IMS Using the SPICE Service Platform. [En línea] 2008. [Citado el: 29 de Marzo de 2008.] https://www.icin.biz/files/programmes/Session7B-3.pdf.

Tekelec. 2007. Service Orchestration and Mediation: Creating a Foundation for Delivering Mixed Services. [En línea] Tekelec, 2007. [Citado el: 2 de Abril de 2008.] http://www.tekelec.com/rcenter/whitepapers/service_orchestration-mediation.pdf.

Telco 2.0. 2009. Telco 2.0. [En línea] 2009. [Citado en: Enero de 2009.] http://www.telco2.net.

Telefónica. 2004. Comunicaciones de Telefónica, Telefónica Investigación y Desarrollo. [En línea] 2004. [Citado el: 16 de Abril de 2008.] http://195.235.92.45/documentos/revista_comunicaciones_i%2Bd/numero34.pdf.

Telefónica I+D. 2005. Las Telecomunicaciones y la Movilidad en la Sociedad de la Información. [En línea] 2005. [Citado en: Abril de 2006.] http://sociedadinformacion.fundacion.telefonica.com/docs/repositorio//es_ES//TelefonicaySI/Publicaciones/telecoymovilidad.pdf.

Tsenov, M. 2002. Application of SOAP protocol in E-commerce Solution. 2002 First International IEEE Symposium, 2002, págs: 59 - 62.

Tikkanen, V. 2006. SOAP/SIP binding. [En línea] Terraventum, 2006. [Citado el: 7 de Marzo de 2008.] http://www.tml.tkk.fi/Opinnot/T-111.5360/2006/SIP.pdf.

Tsang, S., y otros. 2000. SIP Extensions for Communicating with Networked Appliances, Draft. [En línea] IETF, Noviembre de 2000. [Citado el: 8 de Mayo de 2009.] http://www.research.telcordia.com/iapp/draft-tsang-sip-appliances-do-00.txt.

Venezia, C. y Falcarin, P. 2006. Communication Web Services Composition and Integration. IEEE Computer Society, IEEE International Conference on Web Services (ICWS'06), 18 a 22 de Septiembre de 2006, pp. 523-530. 2006.

Villagrá, V. A. y Wesner, S. 2007. AKOGRIMO Mobile Grids: Mobile Dynamic Virtual Organizations. [En línea] 2007. [Citado el: 8 de Mayo de 2009.] http://www.coregrid.net/mambo/images/stories/Events/BIGG/session%205_villagra.pdf.

Zielinski, W. 2007. Competing or Complementary: SIP and Web technologies in migration to NGN. Lessons Learned. Proceedings of the 11th International Conference on Intelligence in Service Delivery Networks. Octubre, 2007.

ANEXO A

ANEXO A

BASE DE CONOCIMIENTO

El presente documento tiene como objetivo exponer de forma general algunas de las alternativas desarrolladas para dar solución a la necesidad de convergencia de los servicios basados en el Subsistema Multimedia IP (IP Multimedia Subsystem, IMS), con el Protocolo de Inicio de Sesión (Session Initiation Protocol, SIP), y los Servicios Web (Web Service, WS), con el Protocolo Simple de Acceso a Objetos (Simple Object Access Protocol, SOAP), por una parte, y al manejo de sesiones en los WS, por otra parte. También se presentan, los conceptos de SIP Servlets y de los Entornos de Ejecución de Lógica de Servicio (Service Logic Execution Environment, SLEE).

A.1. Antecedentes en la Convergencia de servicios IMS (SIP) y Servicios Web (SOAP)

A.1.1. TekSCIM

El Gestor de la Capacidad de Interacción de los Servicios (Service Capability Interaction Manager, SCIM) se define como el elemento de la capa de aplicación de IMS que hace posible la integración de las capacidades de distintos servicios para poder obtener servicios enriquecidos. TekSCIM es la solución SCIM de Tekelec (Tekelec, 2007).

El SCIM, introducido en la arquitectura de IMS del 3GPP (TS 23.002), proporciona una solución para que los operadores puedan utilizar los servicios de mediación y orquestación en el proceso de creación de una arquitectura de servicios flexible, para la prestación de servicios enriquecidos y compuestos, en redes pre-IMS, IMS e híbridas. Está diseñado para realizar la tarea de la interacción de servicios en redes basadas en SIP. Sin embargo, aunque originalmente se imaginó que el SCIM era para redes IMS puras, hoy está claro que su rol necesita extenderse para soportar la mediación a través de múltiples tecnologías.

Hoy en día las redes de telecomunicaciones contienen numerosos sistemas heredados, construidos de acuerdo a estándares y tecnologías que preceden a IMS, con los cuales las redes SIP deben trabajar en conjunto. Adicionalmente a la orquestación de la interacción de los servicios, el SCIM tendrá que dar soporte a toda clase de funciones complejas de mediación y a situaciones únicas en las que se incluyen: uso compartido de servicios a través de las redes, y con distintas tecnologías; comunicaciones de grupo, instantáneas y de colaboración; múltiples tecnologías de acceso; esquemas complejos de direccionamiento y enrutamiento; y un amplio conjunto de protocolos y de interfaces de red.

TekSCIM: creación de una Arquitectura flexible para la Prestación de Servicios

La plataforma de Mediación de Servicios TekSCIM, de Tekelec, está hecha específicamente para la orquestación y mediación de servicios entre redes de múltiples tecnologías. Está construida en una plataforma ACTA de alta densidad, presenta un motor de ejecución potente y basado en reglas que hace posible que los proveedores controlen flexiblemente la interacción y mediación de los servicios al interior y a través de las redes.

TekSCIM conecta las tecnologías, permitiendo que coexistan e interactúen plataformas basadas en el Sistema de Señalización No.7 (Signaling System 7, SS7), la Red Inteligente (Intelligent Network, IN) y la Parte de Aplicación Móvil (Mobile Application Part, MAP) con servicios basados en SIP, para combinar componentes de servicios, prácticamente desde cualquier tipo de red. Con la solución TekSCIM, los operadores pueden:

- Consolidar la mediación y el trabajo conjunto de plataformas IN/MAP con diferentes tecnologías y protocolos.
- Orquestar y manejar la interacción de múltiples aplicaciones para dar soporte a servicios compuestos en redes pre-IMS e IMS.
- Extender los servicios de IN/MAP hacia los dominios de la NGN e IMS, y prestar servicios de próxima generación y basados en SIP a suscriptores TDM tradicionales.
- Mediar múltiples servicios en el dominio IMS para crear una experiencia de usuario con características multimedia enriquecidas.

Casos de uso de TekSCIM

- **Caso de uso #1: Orquestación de Servicios en las Redes Actuales**

  Uno de los retos que enfrentan los operadores en las redes actuales es la provisión de paquetes de servicios compuestos a sus suscriptores. En las redes GSM, un ejemplo es la provisión de ring-back tones personales a suscriptores itinerantes prepago. Las aplicaciones de ring-back tones y de itinerancia prepagada típicamente se alojan en 2 Puntos diferentes de Control del Servicio (Service Control Point, SCP). Cuando se invocan los dos servicios, se crea una composición de servicios y las aplicaciones tienen que trabajar juntas y volverse interdependientes. Para invocar dicha composición de servicios se requieren dos acciones de activación: una para el ring-back tone y una para la itinerancia prepagada. Pero actualmente los Centros de Conmutación Móvil (Mobile Switching Center, MSC) dan soporte a una única acción de activación en lo que se denomina un punto de detección de activaciones.

  Como solución a esto, los operadores que utilicen el TekSCIM pueden lanzar múltiples servicios con una única acción de activación. En este escenario el TekSCIM está desplegado entre el conmutador y la capa de aplicación, como se ve en la Figura A1, y actúa como un SCP virtual. El TekSCIM toma del MSC una única petición de servicio y dirige hacia los SCPs peticiones de múltiples servicios. Luego, adiciona las respuestas y envía una respuesta única al conmutador.

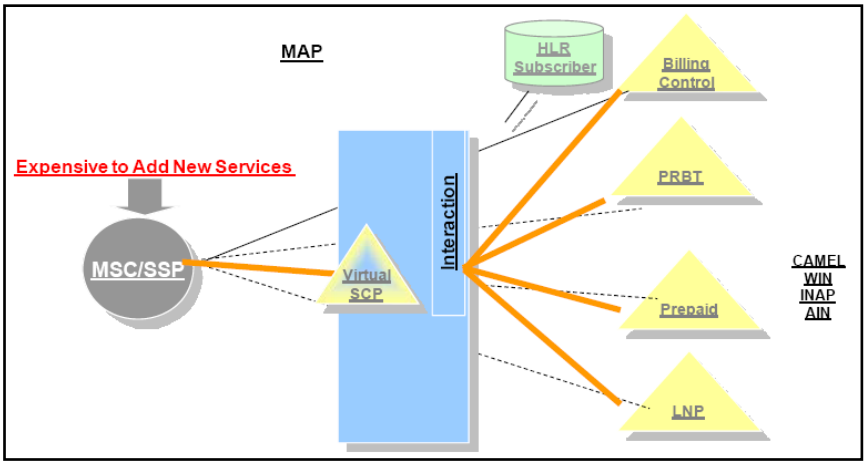


Figura A1. Orquestación de Servicios de Red Inteligente con el TekSCIM

- **Caso de uso #2: Interacción de los Servicios en las redes Pre-IMS/NGN e IMS**

  La provisión de servicios multimedia enriquecidos en redes pre-IMS e IMS representa un reto particular. Para la prestación exitosa de servicios multimedia es crítica la coordinación y el manejo de la interacción de múltiples aplicaciones dentro de una única sesión. Esto requiere de la capacidad de combinar e intermediar tecnologías, características individuales y preferencias personales.

  Por ejemplo, las actuales redes NGN de VoIP contienen una colección incontrolada de elementos tales como softswitches, servidores de medios, servidores de aplicaciones y terminales SIP. Estos elementos ofrecen servicios centrados en la voz y a menudo proporcionan interacción de servicios distribuidos de manera similar a la red cableada heredada. Pero dichos elementos no se construyeron para las especificaciones de IMS, con la funcionalidad centralizada de un SCIM (la cual es requerida para prestar servicios multimedia de valor añadido), en el núcleo de la red, para que maneje la interacción de los datos y recursos. Sin embargo, con el TekSCIM los operadores pueden utilizar la tecnología IMS, basada en SIP, para crear un framework que centralice la interacción y mediación de servicios en el núcleo de la red. Como consecuencia de esto, a los operadores de la NGN/VoIP esta capacidad les permite experimentar los beneficios de IMS en entornos pre-IMS, sin tener que incurrir en el costo de desplegar una red IMS completa.

  Por otro lado, en las redes IMS, el CSCF tiene la habilidad de identificar series de servicios, a partir de la base de datos del suscriptor, y de invocarlos en secuencia. Sin embargo, este elemento carece de la lógica flexible de interacción, y basada en reglas, para manejar la compleja interacción de servicios, y la reutilización de capacidades como la presencia y la localización. Entonces, en el entorno IMS el TekSCIM simplifica la orquestación de servicios mediante la separación del control de la sesión de la capa de servicios, y maneja la interacción entre el CSCF y los múltiples servidores de aplicaciones, como se ve en la Figura A2. En este modelo, el TekSCIM actúa como un servidor de aplicaciones SIP virtual, el cual toma desde la CSCF una única petición de servicio y la dirige a múltiples servidores de aplicaciones. Luego el TekSCIM compara las respuestas provenientes de los servidores de aplicaciones y envía al CSCF la respuesta del tratamiento de la sesión.

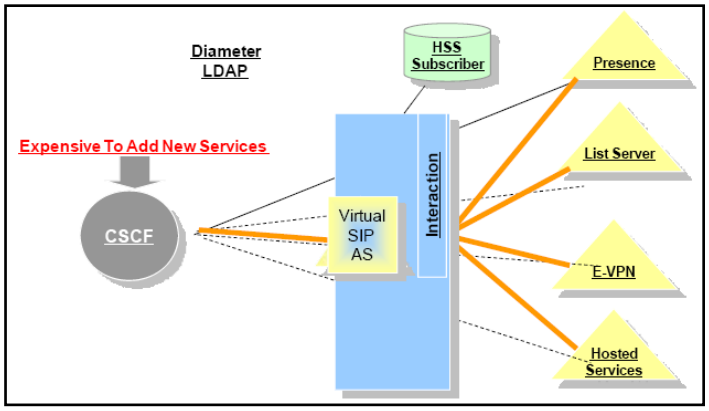


Figura A2. Interacción de servicios en una red IMS con el TekSCIM

- **Caso de uso #3: Mediación e Interacción de servicios entre redes IN y NGN o IMS**

Mientras los operadores hacen que sus redes evolucionen, una consideración importante es cómo hacer que las redes existentes trabajen en conjunto con redes futuras, como IMS, para proporcionales a los suscriptores una experiencia integrada de servicios, sin que tengan que preocuparse por su tecnología de acceso. Otras consideraciones a tener en cuenta, son que los operadores quieren hacer uso de sus inversiones en tecnologías actuales, principalmente aquellas orientadas a servicios, y de esta forma evitar la duplicación de servicios en múltiples dominios, y además, de manera más importante, los operadores necesitan la capacidad de mezclar servicios provenientes de múltiples dominios, para poder crear paquetes únicos de servicios.

El TekSCIM permite conectar redes TDM, NGN e IMS, y proporciona la orquestación y la mediación para hacer posible el trabajo conjunto entre los servidores de aplicaciones basados en SIP y las plataformas de servicios de Red Inteligente, como se observa en la Figura A3. Esto permite que, por un lado, los operadores presten servicios mejorados basados en SIP, tales como presencia, localización, conferencia en la Red Privada Vitual (Virtual Private Network, VPN) e IP, a los suscriptores basados en SS7. Y por el otro lado, los suscriptores IMS tienen acceso a aplicaciones basadas en SS7 como: portabilidad de número, asistencia de directorio y provisión del nombre del llamante.

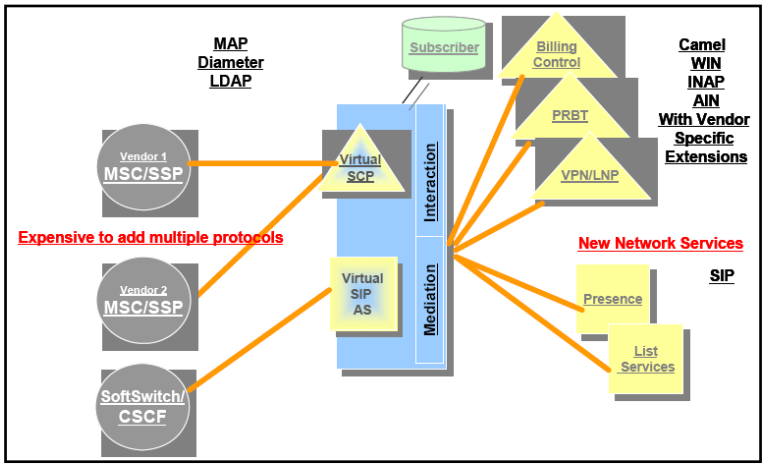


Figura A3. Mediación y Orquestación de servicios entre redes IN, NGN e IMS

No obstante su importancia, las especificaciones del 3GPP contienen muy poca información con respecto a dónde y cómo se debería implementar la función SCIM. Adicionalmente, no hay estándares que definan la integración de servicios y aplicaciones o cómo orquestar e intermediar múltiples servicios para la creación de paquetes de servicios. En gran parte dicha falta de detalles es debido a la controversia de la industria acerca del lugar en la red en el cual debería estar el SCIM.

A.1.2. Plataforma de Servicios para un Entorno de Comunicaciones Innovador (Service Platform for Innovative Communication Environment, SPICE)

Este proyecto, (Tarkoma, y otros, 2008), aborda los retos en la provisión de servicios convergentes, mediante el desarrollo de un framework para el desarrollo rápido de nuevos servicios móviles, en un mundo donde la convergencia de los servicios de telecomunicaciones y los servicios basados en Internet ha adquirido un interés considerable. Además, el objetivo del framework es ocultar las complejidades de un entorno de comunicaciones convergente.

La plataforma SPICE se basa en las Tecnologías de la Información (Information Technologies, IT) utilizadas en el desarrollo de servicios de Internet, pero incluye habilitadores del futuro sistema IMS. SPICE combina diversas tecnologías claves tales como un middleware basado en componentes semánticos, intermediación de servicios y mecanismos de mediación, gestión del ciclo de vida, conocimiento pleno del contexto (Hunor y otros, 2007) y multimodalidad.

Mientras a IMS se la reconoce como parte esencial de la arquitectura de las redes de próxima generación, también es necesario que se mejore el plano de servicios para permitir un aprovisionamiento rápido y fácil de un gran conjunto de servicios. Esto requiere de una plataforma de servicios que reduzca los esfuerzos para la creación y prestación de servicios, en diversos órdenes de magnitud, en términos de los gastos de capital y el tiempo requerido, comparado con los enfoques tradicionales. Actualmente las funciones más interesantes de una plataforma de servicios incluyen:

- *Provisión de identidad*, para permitir que el operador de la plataforma origen del usuario mantenga las identidades de los componentes de los servicios o de los proveedores de servicios de terceras partes, mientras se conserva su seudo anonimato o anonimato.
- *Gestión de la sesión*, para permitir el monitoreo y control de las interacciones con la plataforma de servicios SPICE, dentro de lo que se incluye el uso de varias sesiones de usuario, por parte de distintos servicios, y quizás ejecutados en paralelo.
- *Cobro y facturación*, para permitirle a los usuarios un pago fácil de la conectividad, los servicios y los bienes, mientras se le permite a los operadores que realicen un pequeño cobro adicional de cada servicio.
- *Prestación mejorada de servicios*, por ejemplo a través de tecnologías influyentes, motores de búsqueda, o recomendaciones de servicio con conocimiento del contexto.
- *Personalización y adaptación a situaciones específicas*, para hacer más fácil el uso de servicios móviles y convergentes.
- *Gestión de contenido*. Por ejemplo, contenido generado por el usuario final, gestión de contenido protegida, etc.

- *Itinerancia del servicio*, para permitir que el usuario acceda a los servicios desde la plataforma origen y/o se obtengan los servicios que correspondan o coincidan desde la plataforma visitada.
- *Composición de servicios semántica y automáticamente*, teniendo en consideración el contexto del usuario.

El proyecto SPICE tiene como objetivo permitir este salto en el proceso de prestación de servicios a través de diversas innovaciones. La plataforma SPICE les proporcionará a los operadores las funcionalidades que les permitan el uso convergente de servicios de telecomunicaciones y de IT. Por ejemplo, la combinación de voz, datos y una sesión de video dentro de un servicio, lo cual se conoce como servicios triple play.

Para proporcionar un servicio SPICE los componentes se descubren, federan, combinan, y ejecutan en un entorno distribuido. Los principales avances, que se presentarán a continuación, tienen que ver principalmente con la gestión y autenticación de identidades, la gestión de sesiones, la gestión del conocimiento, y la itinerancia de los servicios; temas fundamentales para la prestación de servicios convergentes. Antes de ello, es necesario que se definan concretamente los Servicios Convergentes como aquellos que utilizan habilitadores del sistema IMS, como por ejemplo el Control de la Sesión, Presencia, Mensajería Instantánea, así como los bloques de construcción de servicios del mundo de las ITs, como por ejemplo los Servicios Web. En ese entorno, SPICE proporciona el medio para conectar estos dos mundos por medio de la convergencia de tecnologías tales como la gestión federada de identidad, gestión de sesiones convergentes, gestión de contextos unificados, y el soporte a la itinerancia de los servicios.

Visión general de la arquitectura de referencia de SPICE (Tarkoma, y otros, 2008) (Hunor y otros, 2007)

Los sistemas de Telecomunicaciones tienen una estructura típicamente en capas. Esto es debido a que se hace una separación de la funcionalidad y las APIs, lo cual promueve un desarrollo, portabilidad y manejabilidad fáciles, con el precio del incremento potencial de los gastos operacionales. Los protocolos se organizan en una pila de protocolos. Arriba de la pila del núcleo de la red se tienen varios middleware, lo cual se ha convertido en una de las piedras angulares o uno de los principios básicos del desarrollo de los sistemas distribuidos modernos. La capa de middleware se puede dividir en subcapas, cada una de las cuales destaca un dominio de funcionalidad específico. Se ha vuelto común el uso de un modelo en capas “no estricto”, y el proyecto SPICE lo sigue, ya que los sistemas middleware cubren muchas áreas de funcionalidades separadas, las cuales no están directamente relacionadas entre sí, como una pila de protocolos de telecomunicaciones. En este tipo de modelos, los componentes de la capa n+2 pueden tener acceso a los componentes de la capa n (o capas menores). En estos casos, el principio de estratificación se utiliza para estructurar el sistema y para indicar niveles crecientes de abstracciones o dependencias.

El proyecto SPICE tiene 4 capas claramente definidas y sus componentes asociados, y se enfoca en los aspectos de núcleo del entorno distribuido de ejecución de servicios, particularmente se tienen: la Capa de Habilitadores y Capacidades (*Capabilities and Enablers Layer*), la Capa de los Servicios Constitutivos (*Component Services Layer*), la Capa de Conocimiento (Knowledge

Layer), y la Capa de Servicios de Valor Añadido (Value Added Services, VAS). Adicionalmente, la plataforma presenta la Capa de Exposición (virtual) (*Exposure Layer*), que proporciona el acceso a la plataforma SPICE.

La plataforma del proyecto SPICE consta de 4 clases distintas de componentes:

- Componentes básicos: son bloques de construcción genéricos que utilizan las características de SPICE.
- Adaptadores de recursos: son componentes básicos especiales, los cuales actúan como proxies para los componentes de los sistemas legados.
- Componentes inteligentes: son los componentes que dan soporte a las interfaces del framework de inteligencia.
- Componentes de servicios de valor añadido. Los componentes básicos e inteligentes se pueden orquestar para crear componentes de VAS. Los componentes compuestos VAS se podrían crear en tiempo de ejecución mediante la utilización de información en tiempo real, provista por varias fuentes, y procesada por los componentes de la capa de conocimiento.

La Figura A4 presenta una visión general de la arquitectura SPICE. La plataforma del terminal se presenta en el lado izquierdo, la plataforma del proveedor de servicios se presenta en el lado derecho, mientras que el Entorno de Creación de Servicios (Service Creation Environment, SCE) se presenta en lo alto de la figura.

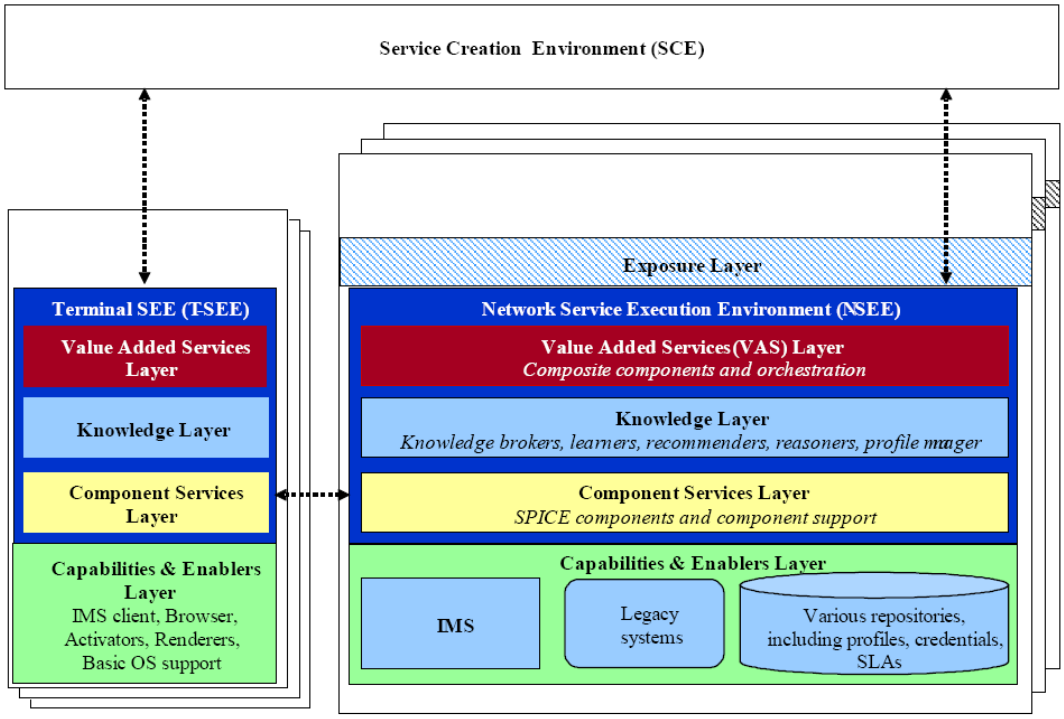


Figura A4. Visión general de la arquitectura SPICE

- **Capa de Capacidades y Habilitadores**

  Esta capa es la responsable de la provisión de las diferentes funciones de soporte que son vitales para la plataforma SPICE. Las funciones de soporte juegan una parte importante al

hacer posible las funciones de núcleo del Entorno de Ejecución de Servicios (Service Execution Environment, SEE). Las funciones de soporte son externas a la plataforma y representan funciones existentes de las redes de telecomunicaciones, como por ejemplo almacenamiento de la base de datos, pilas de protocolos, servicios, señalización básica y mecanismos de control de sesión, y muchas otras funciones. En otras palabras, esta capa consiste en la infraestructura que se encuentra instalada en las redes actuales, por ejemplo la del sistema IMS, la función de soporte OSS/BSS, entre otras.

- **Capa de Servicios Constitutivos**

  La Capa de los Componentes Constitutivos es la que provee las facilidades para el desarrollo, despliegue y gestión del ciclo de vida, basado en componentes. Los "componentes básicos" son bloques genéricos de construcción que aprovechan las características de la plataforma del proyecto SPICE. Los "componentes básicos especiales" son adaptadores de recursos, que actúan como proxies para las capacidades existentes y para los habilitadores de servicios de un operador de red.

  Los componentes básicos, también llamados los componentes SPICE-fied (BEA Systems Iberia, 2007), implementan interfaces específicas para cada servicio e interfaces de gestión. Las interfaces específicas de cada servicio se utilizan durante la composición de los servicios, mientras que las interfaces de gestión se utilizan para controlar el comportamiento y el ciclo de vida de los componentes básicos de la plataforma.

  Además de los componentes básicos esta capa incluye varios componentes middleware, tales como el Gestor del Ciclo de Vida (Lifecycle Manager, LCM), el Registrador (Logger), el Gestor de Suscripción (Subscription Manager, SM), el Supervisor de Componentes (Component Supervisor), y el Supervisor de Disponibilidad (Availability Supervisor, AVSV).

  El modelo y la metodología de componentes dan soporte al despliegue y el aprovisionamiento de servicios a través de varios sistemas. La metodología considera los componentes como un conjunto de objetos de modelamiento que se construyen utilizando notaciones de neutralidad tecnológica, tales como UML y XML. La utilización de este tipo de notaciones le trae a la composición de servicios la robustez y consistencia que tanto requieren.

  La integración con la capa de Capacidades y Habilitadores se hace a través de los adaptadores de recursos, o componentes básicos especiales, los cuales se utilizan para integrar componentes legados con los componentes de SPICE. Los adaptadores de recursos proveen pasarelas para las funcionalidades legadas y al mismo tiempo proporcionan interfaces específicas para servicios de SPICE e interfaces de gestión.

- **Capa del Conocimiento**

  La Capa del Conocimiento define "componentes inteligentes" e introduce varios bloques fundamentales para su construcción, por ejemplo los intermediarios, razonadores y recomendadores de conocimiento. Además, da soporte a la publicación, descubrimiento, entrega y transformación de diversa información como el contexto, el perfil del usuario y la información de presencia. En SPICE, por lo general, a esta información se hace

referencia como conocimiento, y se distingue entre conocimiento de menor y de mayor orden. El conocimiento de menor orden es aquel conocimiento proposicional que se obtiene por medio de la experiencia o de información sensorial. Esto significa que este conocimiento se define como aquella información recolectada de diferentes sensores, dispositivos y fuentes de datos. En contraste, el conocimiento de mayor orden se deriva de las recomendaciones, predicciones, actividades y objetivos.

Esta capa le proporciona a la plataforma de servicios soluciones para un comportamiento inteligente de los servicios, gestión del perfil de usuario, y adaptación de servicios proactiva. Estas soluciones se encuentran agrupadas en distintas familias de habilitadores, que están construidas por encima de un framework común.

La Capa de Conocimiento se realizó como una colección de fuentes de conocimiento distribuidas, sumideros o sinks, e intermediadores. Las fuentes de conocimiento producen información, que es luego consumida por los sinks de conocimiento. La información se entrega ya sea directamente o por medio del empleo del patrón de publicación/suscripción. Los intermediadores son los responsables de la mediación y la transformación de la información, y también de la provisión de interfaces para el descubrimiento de conocimiento. El descubrimiento se realiza ya sea utilizando consultas directas a un intermediador o utilizando el patrón de publicación/suscripción. Los algoritmos de enriquecimiento del conocimiento que se utilizan en esta capa, tales como los algoritmos de aprendizaje y predicción, son conectables y desconectables, y se pueden adicionar nuevas técnicas al sistema.

- **Capa de Servicios de Valor Añadido**

La Capa de Servicios de Valor Añadido es la que facilita la creación de componentes y servicios compuestos a partir de los componentes básicos de SPICE. Para lograr obtener los servicios de valor añadido se necesita de la composición semántica. Los metadatos semánticos de los componentes se encuentran en formato procesable por la maquina y el descubrimiento del conocimiento se utiliza para encontrar los componentes adecuados para la composición. En esta capa, los motores de orquestación son los responsables de asegurar una apropiada sincronización de los elementos de un componente de red compuesto, y además que las interacciones sigan las reglas predefinidas.

Para llevar a cabo una invocación transparente de los servicios constitutivos, los metadatos de los componentes, y las semánticas de las interfaces, se deben desarrollar y publicar en un entorno heterogéneo. La novedad de la metodología es que hace uso de las mejores prácticas de SOA para dar soporte a servicios multimedia, basados en IP, en arquitecturas de red tales como IMS. Esto permite que los diversos interesados tomen parte en el ciclo de vida del servicio, para así tener un entendimiento uniforme de las diversas plataformas de ejecución de servicios.

Entre los habilitadores de esta capa se incluyen la Facilidad para el Descubrimiento (Discovery Facility, DF), responsable del descubrimiento y publicación semántica de servicios; el Intermediador de Servicios (Service Broker, SB), que implementa una arquitectura conectable y desconectable para los motores de composición y ejecución de

servicios; y el Gestor de Itinerancia del Servicio (Service Roaming Manager, SRM), responsable de la composición y prestación de servicios entre plataformas.

- **Capa de Exposición de Servicios**

  A través de la Capa de Exposición los componentes se exponen al mundo exterior. Luego, estos componentes se pueden utilizar y combinar en un entorno multiplataforma para crear servicios compuestos. Para llevar a cabo la publicación, y para anunciar los servicios o componentes, en el sistema SPICE, los entornos de ejecución de servicios de terceras partes pueden utilizar la capacidad de publicación de la plataforma SPICE.

  Entre los habilitadores de esta capa se incluyen el Punto de Ejecución de Políticas (Policy Enforcement Point, PEP), el cual concede el acceso a la plataforma; la Pasarela de Seguridad (Security Gateway, SEG), que hace posible la comunicación entre plataformas, y controla hasta qué grado se abre la plataforma a una plataforma de terceros de confianza; y el componente de Acuerdos de Nivel de Servicio (Service Level Agreement, SLA), que crea y ejecuta acuerdos a nivel de servicio entre los servicios de SPICE y servicios de terceros, sin ninguna interacción humana.

Adicionalmente, SPICE también proporciona un SCE que da soporte a la creación de nuevos servicios, basados en componentes existentes. Se está desarrollando un Estudio de Desarrollo dedicado para desarrolladores profesionales, en torno al Lenguaje de Descripción de Servicios Avanzados SPICE, para los Servicios de Telecomunicaciones (SPICE Advanced Service Description Language for Telecommunication Services, SPATEL), que es el lenguaje de composición de servicios de SPICE.

Arquitectura Funcional (Tarkoma, y otros, 2008) (Hunor y otros, 2007)

La Figura A5 presenta la arquitectura funcional de alto nivel de la plataforma SPICE, la cual puede descomponerse en 4 categorías:

- Plataforma del Terminal
- Plataforma del Proveedor de Servicios
- Entorno de Creación de Servicios
- Proveedores de Contenido y de Servicios de Terceras Partes

La Plataforma del Terminal proporciona solamente una fachada a los servicios, los cuales se manejan y componen a través de varios elementos funcionales en la Plataforma del Proveedor de Servicios. Los Proveedores de Contenido y de Servicios de Terceras Partes pueden contribuir con contenido y con componentes de servicios, en la plataforma, respectivamente, a través de interfaces bien definidas. El Entorno de Creación de Servicios provee herramientas gráficas para que los desarrolladores y usuarios finales de servicios profesionales creen nuevos componentes de servicios, y compongan nuevos servicios a partir de los componentes de servicios existentes.

Los colores de los elementos funcionales visualizan la estructura interna de los paquetes de trabajo del proyecto y al mismo tiempo resaltan los dominios de funcionalidad en los que se centra, particularmente el control de acceso, la gestión de contenido, la gestión del

conocimiento, la gestión del servicio del usuario final, el middleware basado en componentes, el entorno de creación y ejecución de servicios.

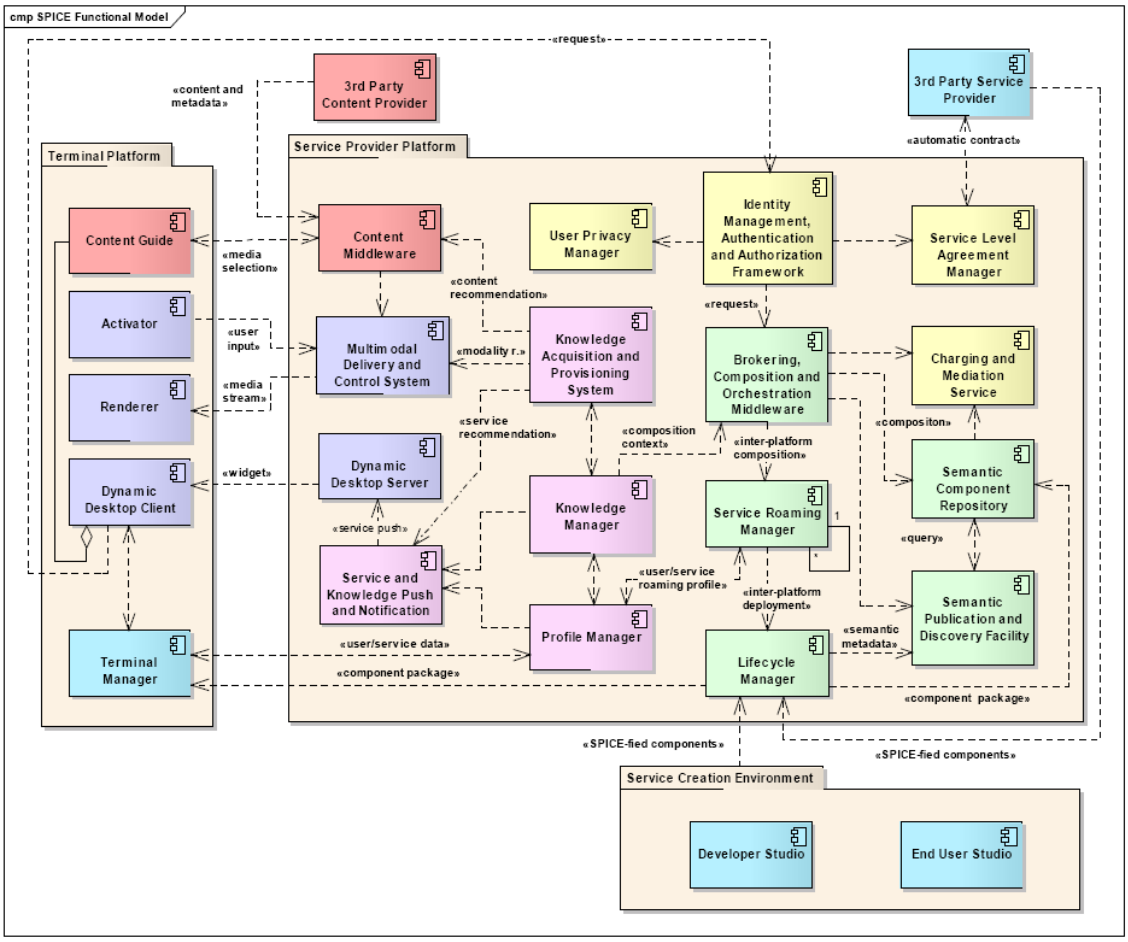


Figura A5. Arquitectura Funcional de SPICE

El **Middleware de Contenido** (*Content Middleware*) tiene en cuenta las recomendaciones de contenido basadas en el contexto y las preferencias del usuario, maneja el contenido generado por el usuario final, y especifica el marco de autorización, donde los múltiples dispositivos que posee el usuario pueden acceder al contenido protegido. Este middleware colabora con los proveedores de contenido de terceras partes, a través de interfaces bien definidas.

El **Sistema de Entrega y Control Multimodal** (*Multimodal Delivery and Control System*) hace posible el traspaso o handover de sesiones multimedia entre los dispositivos del usuario, y dado el contexto del usuario selecciona la modalidad correcta para la sesión multimedia.

El **Escritorio Dinámico** (*Dynamic Desktop*) provee un framework de servicios importantes para los terminales, y maneja un portafolio de aplicaciones dinámico, el cual se ajusta a las preferencias y la situación actual del usuario.

El **Middleware del Conocimiento**, en colaboración con otros componentes del conocimiento, proporciona la modalidad, las recomendaciones de contenido, deduce conocimiento de orden superior (como por ejemplo recomendaciones, predicciones, aprendizajes) a partir de conocimiento de orden inferior (como por ejemplo localización, tiempo actual, preferencias del usuario, presencia del usuario, estado de la batería del terminal, etc.), y también toma en

consideración las políticas de privacidad del usuario. Proporciona una arquitectura de intermediación para realizar la conexión de los sinks de conocimiento, de interés, con las fuentes de conocimiento apropiadas. Este middleware se compone del Sistema de Adquisición y Aprovisionamiento del Conocimiento (Knowledge Acquisition and Provisioning System, KAPS), el Gestor de Conocimiento (*Knowledge Manager*), el Gestor de Perfil (Profile Manager, PM).

El **Middleware de Intermediación, Composición y Orquestación**, implementa un framework donde se componen, orquestan y ejecutan los componentes registrados semánticamente en el momento de la petición, para proporcionarle al usuario final el mejor servicio disponible. El framework les permite a los proveedores de servicios de terceras partes la creación de SLAs con el operador de la plataforma SPICE, sin interacciones humanas, y les permite publicar en la plataforma los componentes llamados SPICE-fied, utilizando el LCM. Para esto último se involucra la publicación de los metadatos en la Facilidad de Publicación y Descubrimiento Semántico (*Semantic Publication and Discovery Facility*) y la instalación del paquete de componentes en el Repositorio Semántico de Componentes (*Semantic Component Repository*).

Un **Framework abierto y controlado de Gestión, Autenticación y Autorización de Identidades** (*Identity Management, Authentication and Authorization Framework*), proporciona acceso multidominio a los recursos y servicios de la red, mediante el mejoramiento de los mecanismos de provisión de identidad, tales como servicios de terceros, Autenticación Única (Single sign-on, SSO), terminal dividido GBA, el cual se explicará más adelante, y por la aplicación de cierta gestión de políticas.

Finalmente, el **Servicio de Cobro y Mediación**, proporciona el cobro en línea y fuera de línea, tanto para servicios simples como compuestos.

A continuación se presentan en detalle los elementos funcionales, así como las características específicas que hace posible la arquitectura.

- **Framework de Gestión, Autenticación y Autorización de Identidad**

  El acceso a los servicios SPICE lo controla un Framework de Gestión, Autenticación y Autorización de Identidad, como se mostró en la Figura A5. Este es un framework distribuido que se encuentra al interior de la plataforma SPICE y forma parte de la capa de exposición, a través de la cual los servicios se exponen al mundo exterior. Provee la característica SSO para que los servicios autorizados autentiquen los usuarios y servicios, sin la necesidad de tener que volverse a autenticar para distintos servicios. En el caso en que los servicios SPICE realicen peticiones de acceso a servicios de terceros, el framework es el responsable de la ejecución de los SLAs relacionados con tales peticiones. El framework de control de acceso también es responsable del control de acceso a los datos confidenciales del usuario.

  El framework es responsable del manejo de los temas relacionados con la Gestión de la Identificación, la privacidad del usuario y en general de la seguridad de la plataforma, dentro de lo que se incluye la autenticación y el control del acceso. La Gestión de la

identidad es una de las tecnologías habilitadoras fundamentales para los servicios convergentes. Ésta debe conectar el mundo de las telecomunicaciones y el de las ITs.

En SPICE todos los usuarios SPICE son suscriptores IMS. Esto permite que los usuarios se autentiquen en la plataforma SPICE utilizando sus credenciales IMS, tales como el Módulo de Identificación del Abonado (Universal Subscriber Identity Module, USIM) y el Módulo de Identificación de los Servicios Multimedia IP (IP Multimedia Services Identity Module, ISIM). Dentro de SPICE los usuarios también pueden tener distintas identidades para los diferentes servicios, y el framework de Gestión de la Identidad (ID-Management, IDM) es el responsable de la gestión de esas identidades.

El proceso de autenticación y la Gestión de la Identidad varían según la red de acceso y el estilo de autenticación utilizada. Cuando se accede a SPICE a través de IMS, por ejemplo, la IDM involucra la utilización de perfiles de servicio definidos en IMS, y cuando se accede mediante un navegador Web, a través de una conexión de Internet normal, se pueden utilizar Liberty Alliance y mecanismos IDM similares. Si se accede a través de la red móvil normal, el IDM utiliza las Configuraciones de Seguridad del Usuario (User Security Settings, USS) de la Arquitectura de Autenticación Genérica (Generic Authentication Architecture, GAA) especificada por el 3GPP.

El documento inicial de la arquitectura de SPICE, para el aspecto de control del acceso, describe 3 alternativas a través de las cuales un usuario puede tener acceso a la plataforma SPICE:

- Utilizar la plataforma SPICE de una manera similar a un servidor de aplicaciones IMS, lo cual implica que se utilicen los mecanismos de IMS para la gestión de la identidad e inicio único de sesión.
- Acceder a la plataforma SPICE a través de HTTPS, haciendo uso de la Arquitectura de Bootstrapping Genérica del 3GPP (Generic Bootstrapping Architecture, GBA) y de la Función de Aplicaciones de Red (Network Application Function, NAF), quien aplica las USS que se encuentran almacenadas con el operador IMS.
- A través de una combinación de GBA y Liberty Alliance, mediante la utilización de credenciales IMS en el procesos de autenticación GBA. Entonces, el acceso del usuario al servicio SPICE se puede ver como un Proveedor de Servicio Liberty Alliance. Esta opción abre la posibilidad de combinar los mecanismos al estilo de Internet, para la federación de la identidad, y también para los servicios de terceros, compatibles con Liberty, por medio de credenciales del mundo de las telecomunicaciones. Tal combinación la implementa SPICE, basado en los conceptos propuestos por el 3GPP para este tipo de trabajo conjunto.

También es posible que para acceder a la plataforma un usuario utilice un dispositivo no IMS, entonces en este caso el dispositivo toma prestadas las credenciales GBA de otro dispositivo, del mismo usuario, con capacidades IMS, por el tiempo que dure el acceso al servicio (se debe recordar que ante todo se pretende que los mecanismos de autenticación de GBA se utilicen con una tarjeta inteligente, o smartcard, que proporcione por lo menos el almacenamiento seguro para el dato secreto compartido). Esto se cumple

por medio del mecanismo de “terminal dividido GBA”. En la Figura A6 se muestra el flujo de trabajo para esta opción.

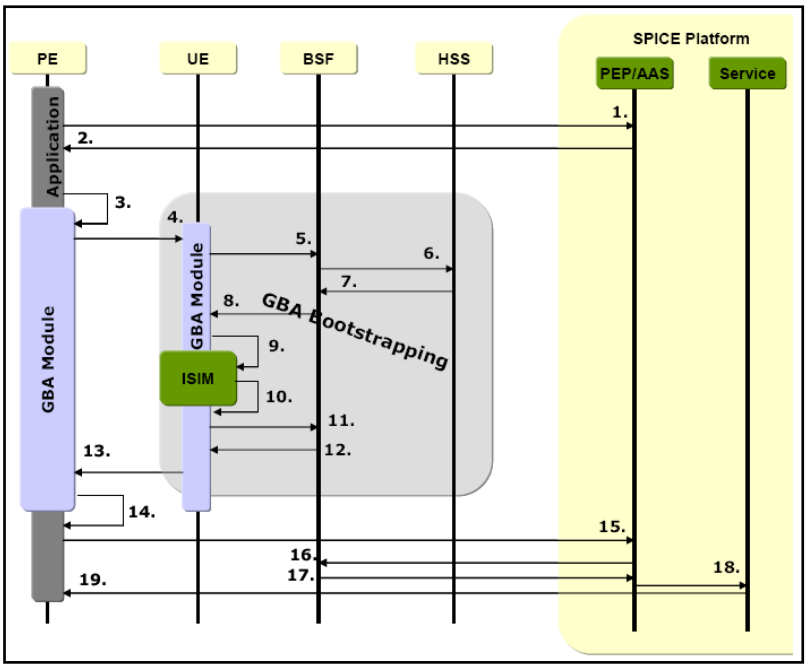


Figura A6. Mecanismo de "terminal dividido GBA"

En el escenario de “terminal dividido GBA”, el Equipo de Usuario (User Equipment, UE) se divide en 2 partes: un Equipo Periférico (Peripheral Equipment, PE) aparte, por ejemplo un computador personal, que tiene la aplicación que utiliza el servicio, y el UE, por ejemplo un teléfono móvil, que tiene las funciones como el proveedor de claves GBA. Los pasos del escenario se representan en la figura anterior.

Los elementos clave de la arquitectura para el control del acceso de SPICE, los cuales se encuentran embebidos dentro de la arquitectura funcional de SPICE, son los siguientes:

- ***Punto de Ejecución de la Política***. Esta es la entidad del sistema que lleva a cabo el control del acceso mediante la realización de peticiones de decisión, y la ejecución de decisiones de autorización. El PEP de SPICE está a nivel de la plataforma (opuestamente al siguiente elemento), es decir, que no es específico a algún componente de servicio. En su mayoría está destinado a ser un interceptor de mensajes: al estar en el flujo del mensaje (por ejemplo como componente de intercepción en un Bus de Servicios Empresariales), éste intercepta los encabezados de los mensajes, le pide al Punto de Decisión de la Política que evalúe las políticas aplicables, y, con base en la respuesta, le permite al mensaje continuar o lo bloquea; en otras palabras, permite o niega el acceso. Cuando se maneja el acceso desde plataformas SPICE foráneas, o componentes de terceras partes, el PEP se apoya en los servicios de la Pasarela de Seguridad (*Security Gateway*) que se verá más adelante.
- ***Punto de Ejecución de Política, específico a un Componente de Servicio*** (Policy Enforcement Point, Service Component-specific, PEP-SC). En los casos donde la información, necesaria para tomar la decisión de control del acceso, no se puede extraer de manera genérica de la petición (cuando se oculta, típicamente en el cuerpo

de la petición, o de alguna otra forma está fuertemente relacionada con la lógica del servicio), el propio componente contiene el PEP-SC, que es quien luego invoca al Punto de Decisión de Políticas (Policy Decision Point, PDP), para tomar la decisión.

- ***El Punto de Decisión de la Política*** es la entidad del sistema que evalúa la o las políticas aplicables, y crea una decisión de autorización. El PDP de SPICE es compatible con el Lenguaje de Marcado Extensible para el Control del Acceso (eXtensible Access Control Markup Language, XACML). Este último es un lenguaje declarativo de políticas para el control del acceso implementado en XML, y es un modelo de procesamiento que describe cómo interpretar las políticas.
- ***Punto de Información de la Política*** (Policy Information Point, PIP). Esta es la entidad del sistema que actúa como una fuente de valores para los atributos. El PDP (o de manera alternativa el PEP) podría utilizar esta entidad para recuperar información adicional, por ejemplo, para un servicio dado, el estado de la suscripción del usuario, el tiempo actual, etc.
- ***Punto de Administración de la Política*** (Policy Administration Point, PAP). Esta es la entidad del sistema que crea una política o un conjunto de políticas. En otras palabras, las políticas son gestionadas (creadas, eliminadas modificadas), y aquellas almacenadas son accedidas a través del PAP.
- El ***Soporte de Autenticación y Autorización*** le da soporte al PEP, mediante la verificación de las credenciales enviadas como parte de la autenticación del usuario final.
- El ***Proveedor de Identidad*** (Identity Provider, IdP) es un proveedor especial de servicios el cual crea, mantiene, y maneja la información de identidad, y provee la autenticación a los otros proveedores de servicios que se encuentran dentro de una federación. El Soporte a la Autenticación y Autorización (Authentication and Authorization Support, AAS) utiliza el servicio en los casos de autenticación Liberty+GBA o sólo Liberty.
- ***Función de Servidor Bootstrapping*** (Bootstrapping Server Function, BSF). Este es un componente clave de la infraestructura GBA (3GPP, 2008), el cual es soportado por un elemento de red, bajo el control de un operador de red móvil. Participa en el procedimiento de *bootstrapping*, en el cual se establece un secreto compartido entre la red y el terminal del usuario final; el secreto compartido se puede utilizar entre las NAFs y los UEs, por ejemplo, para propósitos de autenticación.
- ***Servidor Local del Suscriptor*** (Home Subscriber Server, HSS). Esta es la base de datos principal del usuario, la cual da soporte a las entidades de la red IMS que actualmente manejan las llamadas. Contiene la información relacionada con la suscripción (perfiles de usuarios), realiza la autenticación y la autorización del usuario, y puede proporcionar información acerca de la localización del usuario. Para el *bootstrapping* GBA, el BSF hace al HSS la petición del Vector de Autenticación (Authentication Vector, AV) y de las Configuraciones de Seguridad del Usuario GBA (GBA User Security Settings, GUSS).
- ***Servicio de los Eventos Conexión/Desconexión*** (Logon/off Event Service, LES). Esta es la entidad que distribuye los eventos de conexión y desconexión a los componentes interesados, a través de un mecanismo de publicación/suscripción. De esa forma, este elemento ayuda a los componentes interesados a que inicien y terminen sus propias

sesiones paralelas, las cuales corresponden a una sesión SPICE “toplevel” o de “nivel superior”.

- El ***Gestor de la Privacidad del Usuario*** es una herramienta para manejar los derechos del usuario final. Este elemento decide si los atributos del usuario final podrían ser enviados a terceros o a otros usuarios finales, y lo hace basado en las políticas de permisos de los usuarios, sin que, necesariamente, se tenga que preguntar cada vez al usuario.
- ***Habilitador de Cobro & Mediación*** (Charging & Mediation, C&M). Este elemento soporta la mediación entre las entidades que generan los datos de las cuentas para la facturación de eventos (de servicios al interior de la plataforma SPICE o servicios de terceros) y el dominio de facturación. Éste interactúa con servicios, por un lado con los recursos de red (coleccionando y agregando los datos de las cuentas – datos contables - que éstos producen), y por el otro lado con motores de tarificación, sistemas de facturación etc.
- ***Servicio de Acuerdos de Nivel de Servicio***. Este elemento facilita la negociación automática entre proveedores de servicios y clientes, haciendo posible la creación de contratos sin intervención humana. Esto consiste en hacer publicidad, negociaciones y ejecuciones.
- ***Pasarela de Seguridad***. Esta pasarela implementa servicios de seguridad (autenticidad del origen, integridad, confidencialidad, protección de los mensajes de respuesta) para la comunicación con otras plataformas SPICE (manejadas por otros operadores) o terceros. Este elemento está alojado en el borde de la intranet de los operadores de las plataformas, y todo el tráfico SPICE entre plataformas pasa a través de él.

La Figura A7 ilustra el procesamiento de mensajes que hace el framework de autorización. El PEP intercepta todos los mensajes, luego formula una petición XACML, y la redirige al PDP para obtener una decisión de control del acceso.

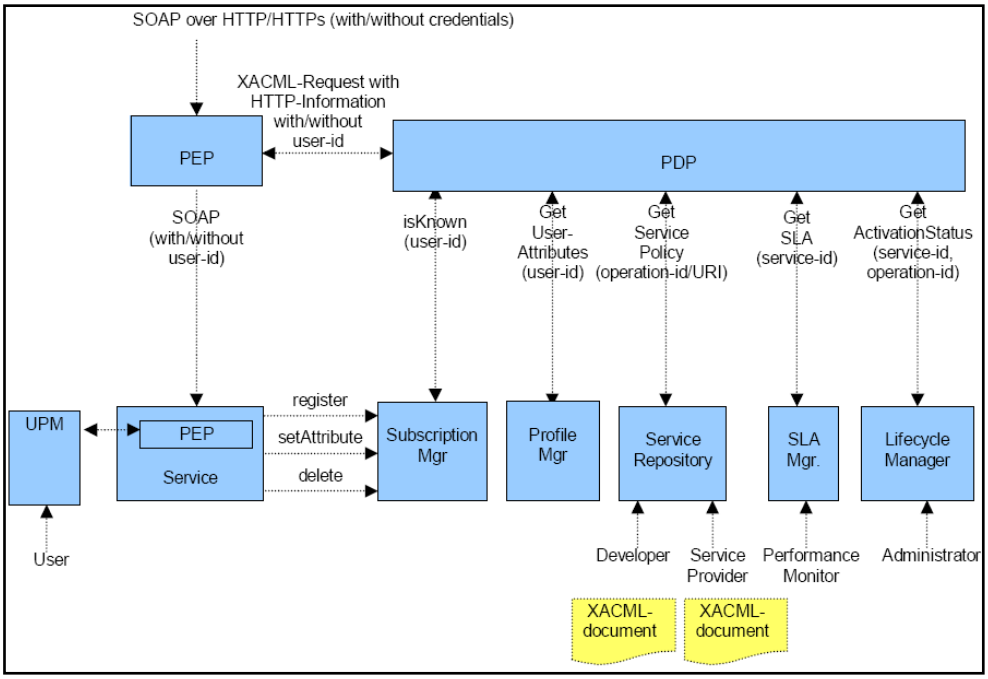


Figura A7. Visión general del soporte al procesamiento y a la gestión de la petición

El PDP interactúa con un número de componentes del sistema, actuando como PIPs, tales como el gestor de la suscripción, gestor de perfil, repositorio de servicios, gestor de SLA, y como gestor del ciclo de vida con el fin de reunir la información necesaria para la evaluación de políticas. Luego se envía de regreso la decisión de autorización desde el PDP hasta el PEP. Si se necesita un segundo nivel de autorización, a nivel de componentes, el PEP global reenvía la petición al PEP a nivel de componente para el procesamiento.

- **Gestión de la Sesión**

Dentro de las plataformas de servicios uno de los temas más importantes es el manejo de las sesiones. La plataforma de servicios SPICE dará soporte a distintos tipos de sesiones, por ejemplo para la ejecución y utilización de servicios, para el acceso a la plataforma, así como para los distintos servicios proporcionados, y también para la comunicación entre el terminal/cliente (con capacidad SPICE) y la plataforma SPICE.

Las sesiones SPICE se establecen entre el cliente y la plataforma SPICE, por ejemplo en la fase de registro del usuario en la plataforma SPICE, y se crea una relación confiable entre el Usuario Final y la plataforma SPICE. El establecimiento exitoso de dicha sesión SPICE también es un punto de activación para el inicio de los servicios de recolección del contexto del usuario y para el de algunos habilitadores básicos, como el servicio de Presencia del Usuario Final. La sesión SPICE será terminada por distintos medios, como la desconexión del usuario, o la cancelación/expiración de la suscripción SPICE del usuario.

Las Sesiones de Servicio del Usuario SPICE son sesiones de señalización/comunicación dedicadas a aplicaciones/servicios específicos del usuario final, las cuales deben tener una sesión SPICE subyacente. Puesto que la convergencia de servicios es uno de los objetivos principales que deberían alcanzar las redes de próxima generación, por ejemplo los servicios triple-play (voz + video + datos) en combinación con la Web 2.0 y los servicios de Internet, se necesita del soporte a la comunicación ubicua, por ejemplo el intercambio de cualquier clase de datos por cualquier clase de aplicación, en cualquier clase de terminal, y utilizando cualquier clase de acceso y núcleo de red. Todos los tipos distintos de aplicaciones requieren de una apropiada gestión de sesión de usuario, incluyendo el descubrimiento de pares y la negociación del formato de los datos, y, comunmente, del control específico de aplicaciones e intercambio de datos.

Los servicios convergentes se caracterizan por los flujos de control que podrían empezar como un intercambio de mensajes SIP y luego activar las interacciones basadas en Internet/Tecnologías de la Información. La plataforma SPICE combina las funciones que se requieren para la provisión de servicios convergentes IT/Telecomunicaciones. Por un lado, a través de los adaptadores de recursos, proporciona acceso al control de la sesión basada en IMS, y proporciona habilitadores basados en Servicios Web para la realización de aplicaciones IT, por el otro. Las Sesiones de Servicio del Usuario SPICE pueden iniciarse en ambos mundos y continuar su ejecución en el otro.

Adicionalmente a las funciones necesarias para controlar los distintos tipos de sesión de la plataforma SPICE, ésta también proporciona las funciones apropiadas para el manejo de la información de estado para esas sesiones. Por ejemplo, la creación y mantenimiento, y

finalmente la eliminación de la información del estado de la sesión soportada dentro del sistema SPICE.

- **Gestión del Conocimiento**

La capa de conocimiento contiene el contexto y los componentes de procesamiento del conocimiento. Tradicionalmente, el manejo de esta información es diferente en el mundo IT y en el de las Telecomunicaciones. En el mundo IT, el contexto se intercambia utilizando los Servicios Web junto con una representación con el Lenguaje de las Ontologías Web (Ontology Web Language, OWL) y el Marco de Descripción de Recursos (Resource Description Framework, RDF). En el mundo de las Telecomunicaciones, el contexto se intercambia tradicionalmente utilizando el Formato de Datos para la Información de Presencia (Presence Information Data Format, PDIF). El objetivo es que los servicios convergentes puedan utilizar la capa del conocimiento con el propósito de adaptar su comportamiento a las distintas situaciones ya sea en del lado IMS o del lado de los servicios de las ITs.

La plataforma SPICE potencia la distribución de la lógica de servicio y de los datos entre las diferentes entidades de la plataforma. Para este propósito, la Capa de Conocimiento provee dos soluciones: el Marco de Gestión del Conocimiento (Knowledge Management Framework, KMF) y el Habilitador del Contexto IMS (IMS Context Enabler, ICE). Su objetivo es proporcionar una infraestructura fácil de utilizar, que estandarice el intercambio y el descubrimiento del conocimiento. Para tener un entendimiento compartido del significado de la información que el KMF y el ICE entregan e intercambian, las fuentes de conocimiento intercambian su información como instancias de una ontología compartida. Para el KMF se escogió un enfoque basado en Internet, el cual se basa en el protocolo GET/POST HTTP, y en SOAP sobre HTTP (Servicios Web). En contraste, la realización del ICE, basado en IMS, tiene un fuerte acoplamiento con la arquitectura IMS y utiliza SIP para su comunicación.

Al utilizar el protocolo SIP y a PIDF, como el formato de los datos del conocimiento, se permite una integración fácil del conocimiento, el cual es fácil de obtener en los habilitadores de servicios OMA/IMS existentes. SIP se utilizará principalmente por dos razones:

1. Para controlar los parámetros de intercambio de las sesiones, en los que se incluyen los conjuntos de datos a comunicar, activación de actualizaciones, frecuencia de las actualizaciones
2. Para ajustar de manera flexible el camino de la comunicación, con base en los cambios de la estructura de la red y la información de contexto disponible.

Dentro de la Capa de conocimiento, en consecuencia a lo anterior, la plataforma SPICE da soporte a tres tipos de fuentes de conocimiento:

- Fuentes de conocimiento basadas en Servicios Web/OWL.
- Fuentes de conocimiento “tradicionales” orientadas a IMS, las cuales intercambian su información con SIP, y utilizando el formato PIDF.

- Fuentes de conocimiento híbridas, que ofrecen ambas interfaces y dan soporte a ambos formatos de datos.

La plataforma SPICE asegura la interoperabilidad entre las fuentes de conocimiento del tipo uno y dos, y a su vez realiza la fusión del mundo de las ITs y el de las telecomunicaciones, mediante la provisión de una pasarela PDIF/RDF, la cual realiza una traducción entre los distintos formatos de datos y protocolos. Adicionalmente a ello, SPICE examina protocolos de intercambio de contexto, basados en SIP y optimizados.

Finalmente en este contexto se tiene a la Ontología Móvil (Mobile Ontology) que es con quien se describe semánticamente el conocimiento, utilizando OWL, tal como lo de define la W3C. El formato de datos subyacente es RDF. Ésta es una ontología de alto nivel para el dominio de las comunicaciones móviles, y está estructurada por medio de un núcleo que describe los conceptos principales de una plataforma de prestación de servicios, y por varias sub-ontologías para dominios detallados. Además, constituye un esquema legible por máquinas, hecho para el uso compartido del conocimiento y el intercambio de información, a través de las personas, aplicaciones y servicios, y cubre dominios relacionados con las comunicaciones móviles. Una de las principales funciones de la Ontología Móvil es hacer que el mundo de las telecomunicaciones sea interoperable con el mundo Web y el de IT, y cuando se tratan distintos lenguajes de ontologías/esquemas, y ontologías y esquemas existentes para ciertos tipos de conocimiento, se necesita aplicar distintas estrategias. La Ontología Móvil es pública para de esta forma extender su utilización a la comunidad investigativa.

- **Entorno de Creación de Servicios**

SPICE proporciona el Entorno de Creación de Servicios, el cual da soporte a la creación de componentes básicos, componentes inteligentes, y servicios de valor añadido. El SCE provee herramientas gráficas para el desarrollo de servicios, así como el entorno para la simulación de los habilitadores de servicios y para la realización de pruebas.

Dos cadenas de herramientas separadas dan soporte a las necesidades de los desarrolladores profesionales de servicios y a las necesidades de los usuarios finales respectivamente.

El Estudio de Desarrollo dedicado para los desarrolladores profesionales se está desarrollando alrededor de SPATEL. Este lenguaje permite la utilización de un enfoque de Arquitectura Orientada a Modelos para la transformación de la idea de un servicio en un servicio ejecutable, a través de una representación independiente de la plataforma.

El Estudio especial para el Usuario Final les permite a los usuarios finales la creación de *mash-ups* de servicios a partir de los servicios SPICE existentes, los cuales son adaptados fácilmente a sus necesidades. Esto significa que los usuarios finales no tienen permitido la composición arbitraria de mash-ups, sino que en lugar de ello tienen una selección de plantillas que estipulan cómo se pueden combinar y parametrizar los servicios. El operador de la plataforma, o los proveedores de servicios de terceras partes, son los que proveen y mantienen estas plantillas. Existen diversas razones por las cuales la composición arbitraria

de servicios de usuario final es desafiante. Una de las razones es la seguridad; porque cada componente introducido en la plataforma debería ser probado y verificado. Otra de las razones es la complejidad de la composición de servicios; idealmente este proceso debería ser simple y fácil de tal forma que el periodo de aprendizaje fuera bastante corto. El patrón "event trigger/action" o "evento de activación/acción" es un patrón importante de diseño para los *mash-ups* de servicios basados en el usuario. En este caso, un motor de monitoreo supervisa los activadores de los eventos definidos, y bajo la detección de un evento las acciones relacionadas se lanzan. Con este mecanismo el usuario final puede construir servicios fácilmente, utilizando componentes de la capa de conocimiento y servicios de comunicaciones.

El proceso de prestación de servicios comprende el conjunto completo de creación de servicios, su puesta a prueba, su despliegue en el entorno de ejecución de servicios (servidor y/o terminal), y luego su provisión a los usuarios finales.

- **Middleware de Intermediación, Composición y Orquestación.**

  El intermediador de servicios es un componente clave de la arquitectura, y es el responsable de la conexión de componentes y de la dirección de mensajes a sus receptores correctos. Cuando la plataforma de servicios recibe una petición, el intermediador de servicios calcula las conexiones de componentes necesarias para servir la petición.

  El modelo de componentes de SPICE le permite a los componentes de las redes adaptarse a las condiciones del entorno, a lo cual se le da soporte a través de la noción de red abstracta de componentes. Estas redes permiten las conexiones entre componentes que no se conocen de antemano, en el momento en que se compone el sistema. Las redes tienen reglas de adaptación que controlan cómo se seleccionan los componentes concretos, y sus conexiones, para adaptarse de la mejor forma al entorno.

  Cuando se utiliza el intermediador para componer servicios, la red de composición de servicios se guarda en el almacenamiento interno del intermediador y se asigna un identificador de sesión a la red. Este ID de sesión se incluye en la petición que procesan los componentes de la red de componentes. Por tanto, el intermediador es el responsable de la gestión pertinente de las sesiones para los componentes compuestos.

- **Itinerancia del Servicio**

  Hoy en día un servicio móvil se ejecuta típicamente en un entorno de ejecución único, como un servidor de aplicaciones IMS. Un servicio compuesto se puede distribuir sobre múltiples dominios y entornos de ejecución de servicios, lo cual le da la naturaleza federada de las NGN, y a los usuarios NGN les da las posibilidades de itinerancia. Esto es debido a que simplemente un servicio en particular no se encuentre disponible nativamente en la plataforma local. Otra razón para la composición de servicios es la localización del servicio o la optimización de la utilización del servicio, debido a la movilidad de terminal o de usuario.

SPICE proporciona un Gestor de Itinerancia de Servicios que previene las barreras típicas de los dominios administrativos, principalmente cuando el usuario es “extranjero”.

El sistema de soporte a la itinerancia de servicios trata los siguientes temas:

- Acceso del usuario a los servicios y la información que escogió en su red local, de origen, o *home*.
- Acceso del usuario a los servicios y la información adecuada (que corresponden al perfil del usuario, las preferencias, y la información del contexto del servicio) desde el dominio visitado.
- La posibilidad del usuario de combinar los servicios, y habilitadores de servicios, de su red de origen con los del dominio visitado, para de esta forma crear nuevos servicios, e inclusive mush-up de servicios.

Con la itinerancia de los servicios el usuario puede migrar a otros dominios distintos al de su red de origen, y luego utilizar los servicios de cualquiera de ellos. Para esto se asume que la plataforma en cuestión tiene acuerdos activos acerca de qué servicios se pueden utilizar en los procesos de itinerancia de servicios. A esto le sigue que el usuario SPICE también debe estar registrado en el dominio SPICE visitado, para que así la itinerancia de los servicios opere apropiadamente.

Vistas de la Arquitectura (Hunor y otros, 2007)

Este análisis se basa en la descomposición del sistema abstracto en capas, componentes funcionales, y vistas dinámicas. Las vistas dinámicas presentan las interacciones típicas entre los componentes funcionales de SPICE, durante la operación de la plataforma. El objetivo de este análisis es presentar una descripción clara y compacta del sistema, sus capacidades y limitaciones, y como se realizaron las implementaciones concretas de los componentes abstractos.

- **Creación de Servicios**

  SPICE da soporte a dos enfoques distintos para la creación de servicios:

  - Creación Profesional de Servicios, utilizando el Estudio de Desarrollo SPICE
  - Creación de Servicios orientada al Usuario Final, utilizando el Estudio de Usuario Final de SPICE

  La siguiente Figura muestra el Proceso de Creación de Servicios y los componentes SPICE que involucra.

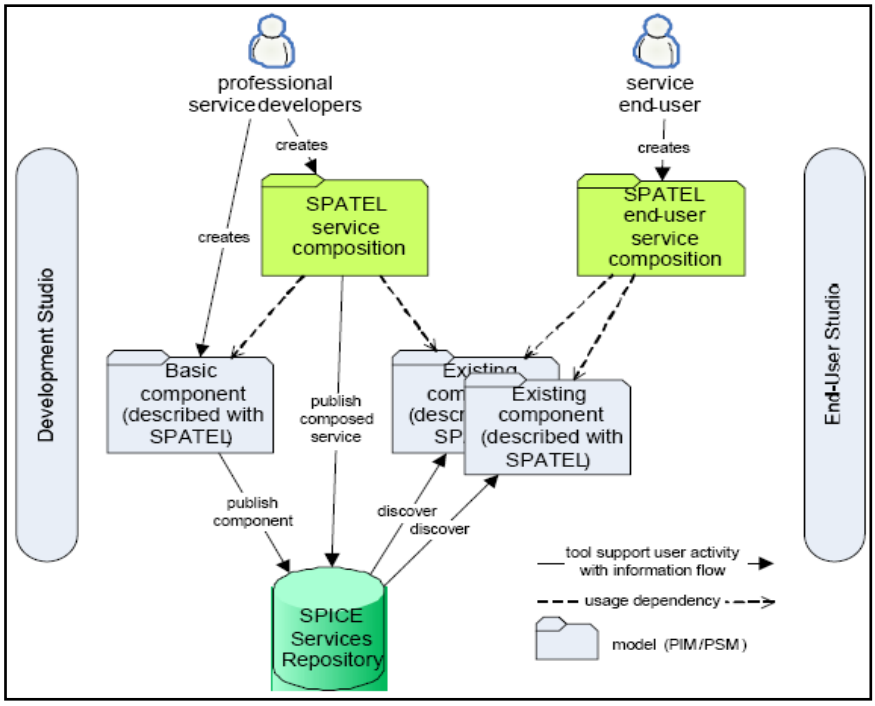


Figura A8. Procesos para la Creación de Servicios

Estudio del Desarrollador

Esta herramienta gráfica permite la creación de una descripción SPATEL de un componente SPICE, por medio de la definición de sus interfaces externas, así como de las interfaces que utiliza. Además, permite el diseño de la máquina de estados que define el flujo de trabajo (workflow) dentro del componente. El Estudio del Desarrollador proporciona las herramientas para:

- Importar/Exportar los archivos del Lenguaje de Descripción de Servicios Web (Web Services Description Language, WSDL)
- Transformar las descripciones SPATEL en bloques de código
- Interactuar con el Motor de Composición Automática
- Describir SPATEL con información Semántica

La Figura A9 da una visión general acerca del Entorno de Creación de Servicios y su enlace con el Entorno de Ejecución de Servicios. En ella se muestra que el Diseñador de Servicios interactúa con el repositorio con el fin de obtener las descripciones WSDL, editar el diseño del servicio, crear la descripción SPATEL, y describirla con la información semántica. El diseñador de servicios también interactúa con el Motor de Composición Automática (Automatic Composition Engine, ACE) para hacer que la composición de servicios esté lista para ser utilizada. Por último, el diseñador de servicios utiliza la transformación de servicios para crear los ejecutables finales que luego se despliegan en el SEE, utilizando el proceso de despliegue y el Gestor del Ciclo de Vida.

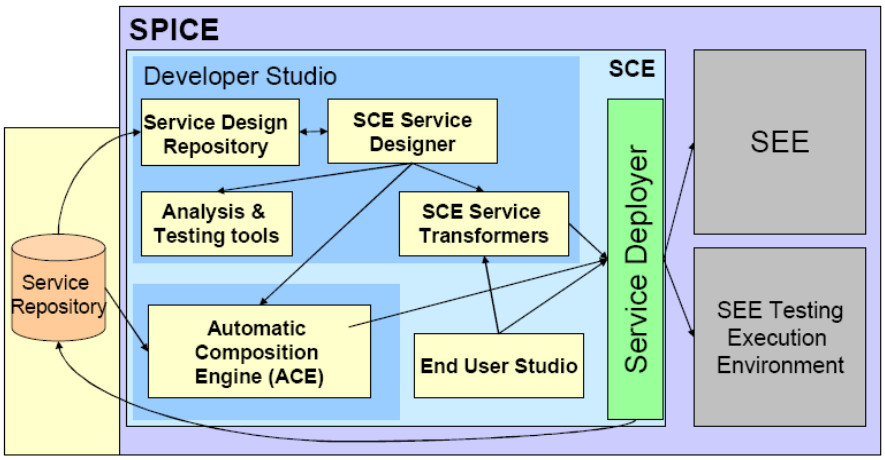


Figura A9. Entorno de Creación de Servicios, y su enlace con el Entorno de Ejecución de Servicios

La Figura A10 muestra la GUI del Diseñador de Servicios SPATEL.

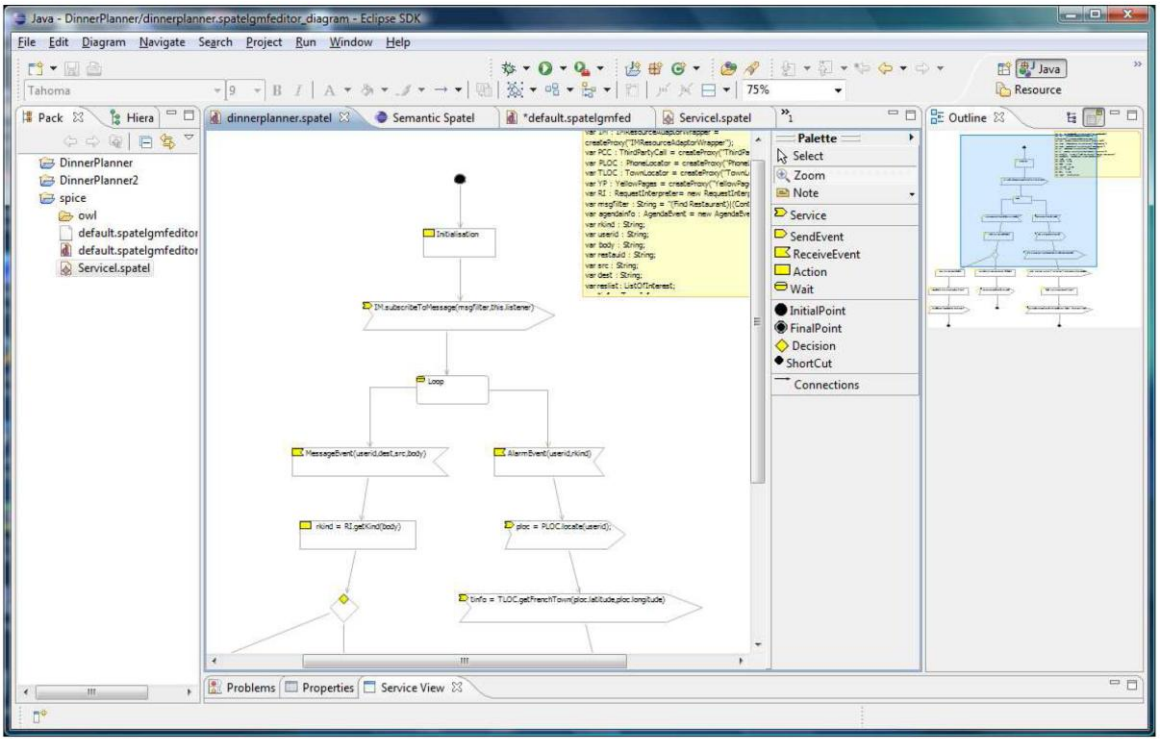


Figura A10. El Diseñador de Servicios SPATEL

Estudio del Usuario Final

Este estudio le permite al usuario final la creación rápida y fácil de servicios. Básicamente este estudio proporciona un editor que permite definir las reglas de activación de las acciones. Con este mecanismo, se crea un paquete de reglas y luego se despliega en el sistema SPICE. La Figura A11 muestra el Estudio del Usuario Final.

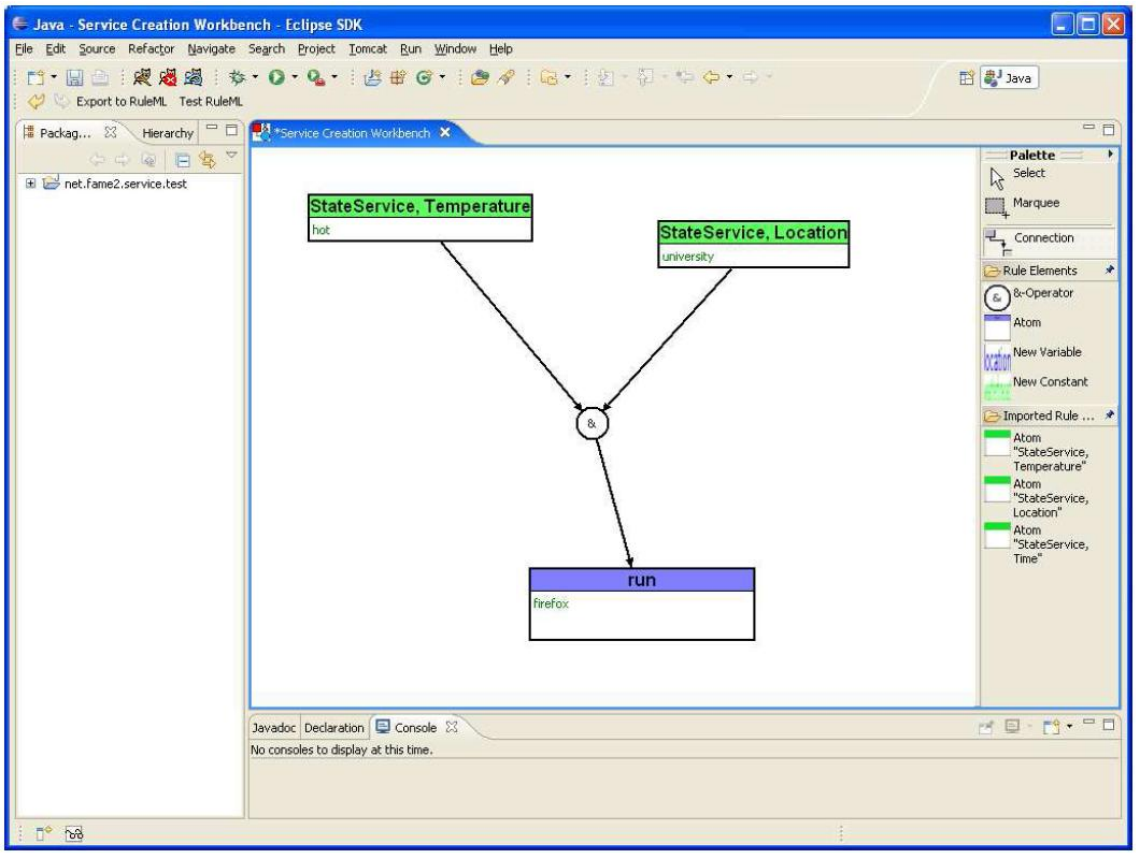


Figura A11. Estudio del Usuario Final de SPICE

- **Entorno de Ejecución de Servicios**

  En la plataforma SPICE los componentes de servicio están alojados en una variedad de motores de ejecución de servicios. El Adaptador de Recursos (Resource Adaptor, RA) es un componente especial que hace la conexión entre la mensajería SIP y el acceso a los Servicios Web. Las diversas implementaciones de un Adaptador de Recursos se pueden llevar a cabo en diversos motores de ejecución. El contenedor de ejecuciones: Entorno Java de Ejecución de la Lógica de Servicio (Java Service Logic Execution Environment, JSLEE) ya proporciona un adaptador de recursos IMS. Por lo tanto, los servicios que se alojen en un contenedor JSLEE pueden actuar como un Servidor de Aplicaciones IMS (IMS Application Server, IMS AS). Un componente especial de SPICE encapsulará el Adaptador de Recursos JSLEE y lo volverá un componente SPICE para que de esta forma los componentes alojados en otros motores de ejecución puedan acceder a IMS. El SEE de SPICE también puede soportar otros contenedores para Adaptador de Recursos, como por ejemplo un Servlet SIP.

  Se debe tener en cuenta que se necesita escribir diferentes Adaptadores de Recursos, para distintos servicios de comunicación. Estos RA especiales manejarán los aspectos específicos de los servicios de los mensajes SIP. Por ejemplo, no es necesario que cada mensaje SIP se envíe a los componentes de la aplicación ya que mediante la utilización de las facilidades de envío del S-CSCF, las peticiones de IMS se envían al núcleo de la plataforma SPICE. Un componente RA transforma la petición SIP a peticiones internas de SPICE basado en WS o en llamadas RMI J2EE. El SEE utiliza los RA que se encuentran

corriendo en el *contenedor de ejecución de JSLEE Mobicent*. Las aplicaciones se pueden llevar a cabo en contenedores de ejecución como el JSLEE, de J2EE, de .NET o del Lenguaje de Ejecución de Procesos Empresariales (Business Process Execution Language, BPEL). Las principales formas de comunicación son las basadas en WS, sin embargo también son posibles las llamadas a tecnologías específicas, como RMI J2EE.

La implementación física de la Plataforma Distribuida de Pruebas, que da soporte a toda la pila de software de SPICE, se lleva a cabo como un conjunto de Entornos de Ejecución de Servicios (SEE) interconectados, e instalados en distintos socios. De hecho, cada uno de estos SEEs es todo un sistema operativo basado en Ubuntu (Linux) y está instalado como una máquina virtual VMware. El SEE contiene servidores de red básicos y plataformas software tales como: DNS, Mono (la implementación de .NET para Unix/Linux), Máquina Virtual de Java, etc., así como el OpenIMS Core, otros Servidores de Aplicaciones y otros Motores de Ejecución (JBoss, JoNaS, Mobicent, Orchestra BPEL, ServiceMix etc.). Toda la plataforma esta previamente configurada y abastecida con scripts especiales, que el proyecto SPICE ha desarrollado, y los cuales permiten una configuración rápida y semi-automática, la monitorización y la administración de toda la pila de software que se encuentra dentro de la Plataforma Distribuida de Pruebas. La forma de instalación que se escogió (como una imagen VMware), junto con el conjunto de scripts, permite distribuir rápidamente las nuevas versiones del SEE y conectarlo perfectamente a la plataforma distribuida, sin la necesidad de cambiar, en varios lugares dentro del SEE, los ajustes de la configuración básica (tales como: el nombre del dominio IMS, la configuración IP, etc.).

Trabajo conjunto de IMS y SPICE (Hunor y otros, 2007)

IMS es la arquitectura aceptada para las Redes de Próxima Generación. Por lo tanto, es esencial que se logre un trabajo conjunto y óptimo entre SPICE e IMS. Por las directrices de la arquitectura SPICE se consideraron principalmente dos niveles distintos de integración entre SPICE e IMS como candidatos apropiados: el fuerte y el bajo acoplamiento con IMS. Estos candidatos se resumen en la Tabla A1.

Tabla A1. Resumen de las estrategias de integración de IMS

	Fuerte acoplamiento	Bajo acoplamiento	Enfoque seleccionado
Gestión de la Identidad	Subscripción y registro IMS	IMS u otra solución	Se necesita subscripción IMS, pero no se necesita un registro activo
Acceso a la Plataforma	SIP, HTTP	HTTP, SOAP, RPC	SIP, HTTP, SOAP
Servidor de Aplicaciones	AS IMS	Un amplio rango de servidores	Se prefiere AS IMS, pero no de manera exclusiva
Habilitadores de servicios	OMA/IMS	Servicios de Internet	Se prefiere OMA/IMS, pasarelas para la interoperabilidad

La estrategia de integración que se eligió en SPICE es una combinación del fuerte y del bajo acoplamiento. La estrategia seleccionada requiere de una suscripción IMS, pero no requiere un registro activo de IMS, lo cual permite efectivamente la utilización de Servicios Web y otras tecnologías para acceder a la plataforma. El Servidor de Aplicaciones IMS es el entorno

preferido para la ejecución de los servicios, pero el enfoque también permite servidores que no se basen en IMS. Además, se prefieren con prioridad los habilitadores de servicios OMA/IMS a otros habilitadores.

A.1.3. SOAP y SIP

SOAP es un mecanismo de transporte neutral para el intercambio de mensajes. Esto significa que cualquier protocolo de transporte o de la capa de aplicación puede transportar un mensaje SOAP, en cuanto cumpla el conjunto formal de reglas que se definen para ello; de esta forma, es posible transportarlo a través de HTTP o SIP.

SOAP/SIP Binding

Cuando se utiliza a SOAP por encima de SIP, es decir en el plano del usuario (datos), se hace posible la transmisión de los mensajes SOAP al interior del cuerpo de un mensaje SIP (Gehlen, y otros, 2006). Esta alternativa corresponde al caso predeterminado de transmisión de mensajes SOAP al interior de un mensaje HTTP, fue desarrollada por la empresa Ubiquity Software y propuesta a la IETF. En ella se introduce un nuevo método SIP: "SERVICE", con el cual es posible llevar mensajes SOAP como carga útil de un mensaje SIP (Deason, 2001). Esta propuesta ya se utiliza como un método aceptado en muchas situaciones, y a continuación se presenta un ejemplo de uso (Deason y Ubiquity, 2000):

```
SERVICE sip:broker.ubiquity.net SIP/2.0
To: sip:broker.ubiquity.net
From: sip:proxy.ubiquity.net
Call-ID:648324@193.195.52.30
CSeq: 1 SERVICE
Via: SIP/2.0/UDP proxy.ubiquity.net
Content-Type: text/xml
Content-Length: 381

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
            SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encodi
      ng/">
      <SOAP:Body>
            <m:SetCreditStatus
      xmlns:m="http://www.ubiquity.net/sipservices">
                  <m:user>sip:jo@ubiquity.net</m:user>
                  <m:status>super</m:status>
            </m:SetCreditStatus>
      </SOAP:Body>
</SOAP:Envelope>
```

- **Casos de uso** (Tikkanen, 2006)**:**

- ***Servicio de Traducción para Mensajería Instantánea***

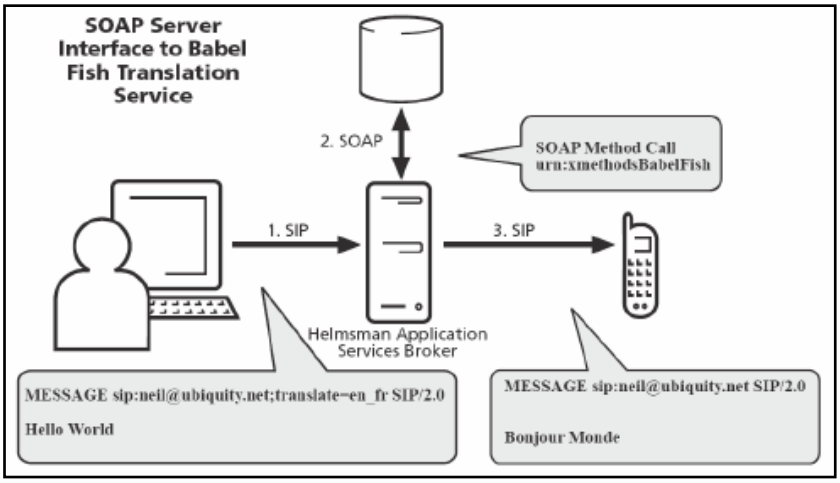


Figura A12. Servicio de Traducción para Mensajería Instantánea

El proveedor del servicio de mensajería instantánea utiliza Altavista's babelfish para traducir los mensajes de texto: James le envía a Neil un mensaje.

```
MESSAGE sip:neil@ubiquity.net;translate=en_fr SIP/2.0
To: sip:neil@ubiquity.net
From: sip:james@ubiquity.net
Call-ID: fd835c@193.195.52.229
Via: SIP/2.0/UDP 193.195.52.229
Contact: sip:neil@193.195.52.229
CSeq: 1 METHOD
Content-Language: en
Content-Type: text/plain
Content-Length: ….

Soap –message..
```

El manejo del transporte del mensaje básico se lleva a cabo con SIP. El servicio de traducción se maneja como una aplicación de WS de terceras partes al cual se accede mediante mensajes SOAP. El servidor puede traducir el mensaje entero y enviarlo a los destinatarios. Mediante esta técnica, personas de diferentes lenguas podrían comunicarse.

- ***Servicio de Entrega de Regalos***

  Cuando un regalo se ha despachado, la persona que lo envía recibe un mensaje instantáneo el cual incluye una URL. Esta URL es un servicio de petición de llamada que abre una conexión con el destinatario. SIP escoge el dispositivo adecuado del destinatario, y SOAP le indica al servicio los canales que se van a utilizar.

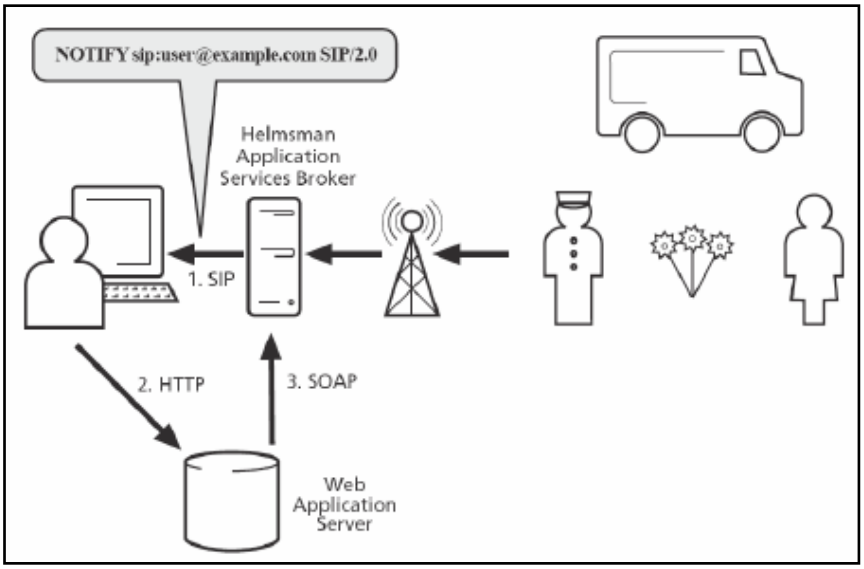


Figura A13. Servicio de Entrega de Regalos

- ***Servicio compra de videos en línea***

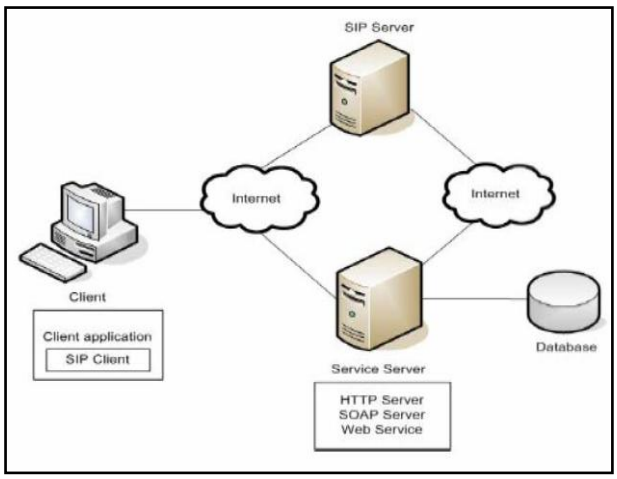


Figura A14. Servicio compra de videos en línea

SIP inicia la sesión que se va a utilizar para la comunicación entre el cliente y los servidores. La comunicación se basa en la conexión permanente de SIP y no necesita almacenarse en las peticiones.

1. La Conexión SIP se establece entre el servidor SIP y el cliente.
2. Se crea un ID único para la sesión.
3. El cliente y el servidor inician la comunicación con SIP/SOAP.
4. El servidor SIP y el servidor actual de e-commerce pueden utilizar el mismo ID.
5. La conexión se elimina, borrando el ID SIP correspondiente.

- ***Sistema de Servicios Web E-Banking***

El usuario establece una sesión con el servidor SIP, transfiere dinero entre dos bancos y termina la sesión. El protocolo SIP se utiliza para gestionar las sesiones entre todas las partes, y puede manejar las características de seguridad.

En algunos de los escenarios reales en los que se aplicó esta alternativa se demostró, por ejemplo, que SIP se puede utilizar para escoger el dispositivo adecuado del destinatario y SOAP le puede indicar al servicio los canales que se van a utilizar.

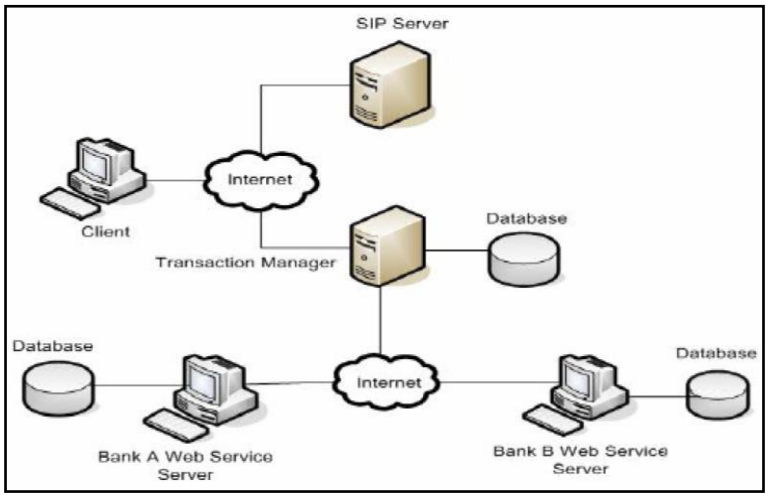


Figura A15. Servicio compra de videos en línea

SIP + DO (López y Arcas, 2007)

Otros intento de aprobación de métodos similares, es el caso del método SIP del tipo "DO" (Tsang, y otros, 2000), propuesto por la empresa Telcordia Technologies, y que se incluye dentro del proyecto HomeSip (IMS Laboratory, y otros, 2006), para la utilización del protocolo SIP como protocolo de control de una casa domótica, nunca se han llegado a aceptar alguna de éstas propuestas.

Para desarrollar el nuevo mensaje del tipo DO, se modifico la implementación ofrecida por NIST de Jain Sip, en esta nueva implementación se añadió un nuevo tipo de mensaje, y se genero una lógica de respuestas a éste mensaje. A continuación, en la Figura A16, se muestra un ejemplo de mensaje SIP de tipo DO:

```
DO sip:Prueba@192.168.48.209:5080 SIP/2.0
Call-ID: 4f242473e77d1f9f6b4705cafad46c6d@192.168.48.209
CSeq: 5 DO
To: <sip:Prueba@192.168.48.209>;tag=12345
From: <sip:3@192.168.48.124:5060>;tag=4321
Max-Forwards: 68
Via: SIP/2.0/UDP 192.168.48.124:5060;
Record-Route: <sip:192.168.48.124:5060;lr>
Content-Type: text/xml
Content-Length: 54

<action type="SELECT"><user value="2"/></action>
```

Figura A16. Mensaje SIP de tipo DO

Tal y como se observa en la figura, el mensaje DO, contiene las cabeceras genéricas de un mensaje SIP, el "content type" indica el tipo MIME del contenido del mensaje, y dentro del cuerpo del mensaje se puede observar un pequeño documento en XML que indica la acción que se debe ejecutar en la Unidad de Control Multipunto (Multipoint Control Unit, MCU). La implementación realizada de este mensaje permite el uso de SIP como contenedor tanto de

XML como de objetos serializados con SOAP. Finalmente se selecciono el contenido XML, para facilitar el desarrollo de la aplicación, aunque durante el desarrollo de esta siempre se tuvieron en cuenta las necesidades de la serialización con SOAP, para que en un futuro sea posible evolucionar hacia esa tecnología sin problemas.

Cuando la MCU recibe el mensaje del tipo `DO`, Figura A17, comprueba sus cabeceras y su contenido, una vez realizadas todas las acciones necesarias, contesta al cliente que genero el mensaje con una respuesta del tipo `200 OK`; esta respuesta no debe ser confirmada por el otro extremo, en caso de pérdida de alguno de los dos mensajes (petición o respuesta), el cliente que genera la acción, es el encargado de reenviar el mensaje del tipo `DO` de nuevo.

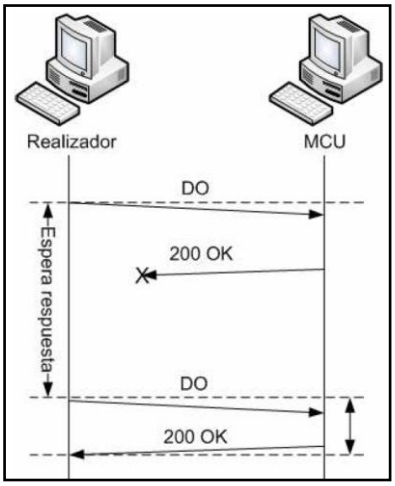


Figura A17. Intercambio de mensajes de tipo DO

El mensaje XML generado por el Realizador representa el tipo de acciones que este puede ejecutar sobre la MCU. A continuación se presenta un ejemplo, en XML, de una acción generada por la MCU:

```
<action type="BAN"><user value="1"/></action>
```

El atributo "`type`", representa el tipo de acción, y "`value`" representa el usuario al que va a afectar la acción. El atributo `type`, puede tomar como valor cualquiera de las tres acciones posibles ("`SELECT`","`BAN`","`BYE`"), y en el atributo "`value`" una cadena de caracteres que representa el nombre del usuario. Por ejemplo en la figura anterior la acción que se demandaba era la de vetar el acceso a la MCU, para el usuario identificado como `1`.

No obstante la importancia de estos nuevos método SIP, nunca se han llegado a aceptar y estandarizar sus propuestas.

A.1.4. Creación de Servicios Web P2P y móviles utilizando SIP

Los servicios de telecomunicaciones son naturalmente servicios Par a Par (Peer to Peer, P2P), por ejemplo las sesiones de llamada de telefonía/video, o el intercambio de mensajes. Pero estos son apenas un subconjunto pequeño de los servicios P2P. En un futuro será normal, para los usuarios de terminales móviles, las aplicaciones en las que se comparte una gran diversidad de información, como el álbum personal de fotos, una pizarra virtual, o juegos de múltiples jugadores. En general, se podrían compartir servicios arbitrarios a través de la lista de

contactos de un usuario. Para hacer posible tales aplicaciones, los desarrolladores de aplicaciones necesitan un soporte robusto a fin de hacerle frente a: las características de movilidad de los usuarios, las redes y lo cambiante de sus condiciones, la gestión de sesiones de aplicaciones fluidas, los entornos y contextos de usuario cambiantes, la variedad de tipos de dispositivos y sistemas operativos, y la variedad de tipos y de protocolos de red.

El framework de Servicios Web Móviles que se propone en (Gehlen, y otros, 2006) oculta a los desarrolladores de aplicaciones la complejidad de todo el sistema y de la provisión de los servicios, y toma en cuenta, de manera transparente, la lista de desafíos que se nombraron anteriormente. IMS se utiliza para la gestión de la sesión y de la movilidad con el fin de permitir sesiones completamente transparentes, incluso si cambian la Red de Acceso Radio (Radio Access Network, RAN) o el Terminal actual utilizado. Por ejemplo, un usuario jugando ajedrez con un amigo desde un PC puede cambiar a su teminal móvil sin tener que reiniciar el juego, manteniendo la sesión de ajedrez corriendo mientras se cambia su localización. De manera adicional, los nuevos tipos de redes y condiciones no provocan la terminación de una sesión (se aplica soporte a la movilidad, roaming y enrutamiento de los mensajes). Además de esto, dependiendo del modelo de negocio, IMS hace posible la facturación de servicios arbitrarios, ya sea en línea (modo conectado) o fuera de línea (modo desconectado).

Los desarrolladores de aplicaciones se pueden soportar en el concepto de las Arquitecturas Orientadas a Servicios y en las tecnologías de WS para poder concentrarse más en la aplicación que en tratar de entender y manejar los dispositivos o las características específicas de la comunicación. Si se utilizan estas tecnologías, manipular una aplicación distribuida es tan simple como tratar un componente software local. Los WS se proveen y publican como servicios convencionales, por ejemplo una oficina postal o un traductor. Entonces, el desarrollador de aplicaciones tiene la capacidad de buscar e examinar servicios de su interés mediante el acceso a un registro, tal como cualquier persona podría estar consultando en las páginas amarillas. El framework descrito se centra en los puntos de contacto de IMS y las tecnologías de WS, su interworking, y sus mejoras. Estas adaptaciones permiten una integración transparente de los WS Móviles en la arquitectura IMS (combinación de SOAP y SIP) y una comunicación de WS mejorada, utilizando protocolos alternativos de enlace, a través de redes móviles.

Un aspecto importante de los WS móviles es su desempeño, dado que las redes móviles cambian frecuentemente y en general ofrecen mala Calidad de Servicio (Quality of Service, QoS). Para las aplicaciones de WS Móviles el parámetro de QoS más importante es la latencia de la Llamada a Procedimiento Remoto (Remote Procedure Call, RPC), el cual tiene que reducirse hasta un mínimo para garantizar aplicaciones interactivas fluidas.

Servicios Web Móviles (Mob-WS)

Este término no se encuentra definido claramente y se ha utilizado con diferentes significados (en diferentes contextos y dominios). En este trabajo, el término se utiliza en cualquiera de los casos donde redes y dispositivos móviles se involucren en las interacciones de los WS. Cuando se ejecutan los WS desde dispositivos móviles se deben tener en cuenta pocos cambios en esta tecnología, por ejemplo el mejoramiento de la comunicación de WS a través de redes móviles, utilizando enlaces con protocolos alternativos. El hecho de proporcionar Servicios Web y

Servicios Web P2P Móviles demanda esfuerzos y cambios adicionales en las tecnologías básicas de los WS; por ejemplo el soporte a la movilidad para facilitar la movilidad y la gestión de la sesión de los Mob-WS para utilizarlos en un modo P2P. En este caso la cooperación con IMS es fundamental.

Arquitectura del protocolo Mob-P2P-WS

Desde el punto de vista de los protocolos pueden surgir dos arreglos posibles de combinación de SOAP y SIP. SOAP se puede utilizar por encima de SIP, o en paralelo en la misma capa. En este sentido, cuando se utiliza a SOAP por encima de SIP, es decir en el plano del usuario (datos), se hace posible la transmisión de los mensajes SOAP al interior del cuerpo de un mensaje SIP. Esta alternativa corresponde al caso predeterminado de transmisión de mensajes SOAP al interior de un mensaje HTTP (Hypertext Transfer Protocol), pero desde el punto de vista de los protocolos no existe diferencia, ya que HTTP y SIP tienen sintaxis y servicios similares.

SIP se define como un protocolo de señalización en la capa de aplicación, por lo que es posible su utilización en el plano de control (señalización) en paralelo con SOAP, que a su vez se encuentra en el plano del usuario (datos). La separación estricta entre los planos de usuario y de señalización tiene ventajas con respecto al diseño de los protocolos, al diseño del software de comunicación, y al desempeño (por ejemplo, se reduce la carga de la red con la infraestructura SIP). En este sentido, y como se muestra en la Figura A18, en el Plano del Usuario se pueden transmitir los datos específicos del usuario de la aplicación a través de SOAP, por encima de los diversos protocolos de Internet subyacentes, y en el Plano de Control (Señalización), se puede utilizar SIP para transmitir los mensajes de señalización de la capa de aplicación.

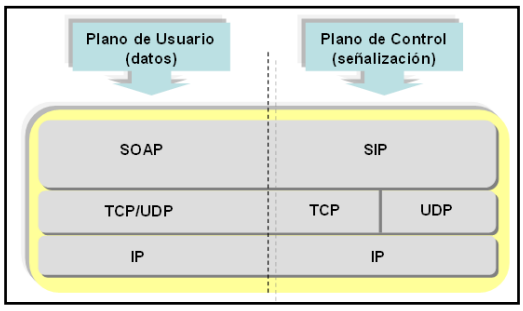


Figura A18. Plano de Usuario y de Control del Servicio Web Móvil

A.1.5. WSIP: Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP

La combinación de SIP con los WS proporciona un nuevo paradigma de comunicación convergente. Otra alternativa existente en esta área es el Terminal WS/SIP para Comunicaciones Multimedia y Multimodales sobre IP (Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP, WSIP).

En (Liu, y otros, 2004) se presenta un enfoque para la convergencia de servicios de comunicaciones sobre IP, basado en el concepto de WSIP (Web Service SIP). En este enfoque cada nodo WSIP es tanto un nodo SIP, que se comunica en el mundo de SIP a través de esta

señalización, y también es un nodo de WS (SOAP), que proporciona un entorno nativo y genérico para la integración de servicios, obteniendo de esta manera un enfoque para la unión de la comunicación basada en SIP con los WS.

La combinación de SIP con los servicios Web proporciona un nuevo paradigma de comunicación convergente. Este permite el descubrimiento y el enlace dinámico de servicios, el cual integra a SIP como un componente un la transacción empresarial. La clara separación del entorno de integración de servicios de la señalización SIP mantiene la simplicidad y la eficiencia del actual protocolo SIP, al tiempo que se aprovecha la potencia de ambos métodos para los servicios de comunicaciones basados en IP. La naturaleza ubicua de la integración de servicios mediante los WS proporciona una solución disciplinada para mantener y actualizar un gran número de nodos SIP. Esto permite el monitoreo, la reparación, y la actualización dinámica de los Agentes de Usuario (User Agent, UA) SIP, lo cual es esencial para el éxito de la VoIP utilizando SIP.

Con el fin de integrar a SIP y los servicios de comunicaciones convergentes, existe una crítica necesidad de un entorno de integración de servicios extensible, el cual permita que las aplicaciones integren a SIP como parte de los procesos de las transacciones empresariales. WSIP tiene en cuenta la separación de la señalización y del medio de transmisión que utiliza SIP, y la lleva al siguiente nivel: separación de la integración de servicios de la señalización. Al separar la señalización de la transmisión de medios, SIP tiene la ventaja de ser un protocolo liviano, de nivel de aplicación, para la configuración y terminación rápida de llamadas, porque no se transmite la multimedia a través de los mensajes de configuración de la llamada.

Para integrar a SIP en las transacciones empresariales, y a través de XML, se han propuesto diversas alternativas. Una de ellas consiste en embeber el control/respuesta del servicio, basado en XML, dentro de los mensajes originales de petición de SIP, tales como INVITE, INFO, SUBSCRIBE, NOTIFY, y como parte de los cuerpos de los mensajes. Este enfoque pragmático encuentra varios problemas, debido a que en el protocolo SIP se carece de las semánticas que den soporte a tal uso. Otro intento consiste en embeber el control/respuesta del servicio, basado en XML, en el Protocolo de Descripción de la Sesión (Session Description Protocol, SDP), como parte del INVITE. Sin embargo, SDP en más una sintaxis de descripción de sesiones que un protocolo, no proporciona un rango completo de capacidades de negociación para el control/respuesta de servicios, no está diseñado para extenderse fácilmente, y las reglas de análisis sintáctico que posee son estrictas. Adicionalmente, esta alternativa saca a SDP de su uso actual en el mensaje INVITE, a lo cual también se suma que el UA SIP receptor puede ignorar el SDP del INVITE y responder con su propio SDP para la configuración de la llamada.

Por lo anterior, si se trata de embeber directamente el control/respuesta de los servicios dentro del protocolo actual de señalización, esto tiene algunos inconvenientes. Primero, se desvía el uso actual del protocolo SIP, y se sale del propósito de que sea un protocolo liviano a nivel de aplicación, el cual separa la señalización de la multimedia. Este enfoque interferiría con el protocolo SIP original. Debido a la falta de una estructura extensible y estándar basada en XML, existe un serio peligro de introducir inconsistencias entre los UAs SIP, y la interoperabilidad de servicios puede estar en un verdadero riesgo. Segundo, no existe un entorno de integración de servicios estándar para los nodos SIP. Una consecuencia de

embeber directamente en la señalización SIP el control del servicio, consiste en que una aplicación encargada de manejar un nodo SIP tiene que contener en ella misma el protocolo como tal, debido a que todos los controles de integración de servicios se comunican a través de la señalización del control de la llamada. Esto no siempre puede ser factible e incurriría en una sobrecarga innecesaria, especialmente para las aplicaciones basadas en XML.

El enfoque de WSIP proporciona una interfaz unificada para la integración de servicios con SIP. La utilización de SOAP en WSIP proporciona una estructura extensible, y bien formada, para el control remoto de servicios y el intercambio de datos. La unificación de SIP con los WS proporciona un nuevo paradigma de comunicaciones convergentes. Este permite el descubrimiento dinámico de servicios, su reparación/actualización, y enlace, el cual integra a SIP como un componente en las transacciones empresariales. Esta clara separación del entorno para la integración de servicios y la señalización SIP, mantiene la simplicidad y la eficiencia del protocolo actual, y aprovecha la eficiencia de ambos métodos en los servicios de comunicaciones convergentes sobre IP.

La idea principal detrás del enfoque de WSIP es no complicar más el protocolo de señalización SIP, sino exponer en el mundo de los WS un UA SIP como un nodo SOAP. Al hacerlo, se separa la señalización del entorno de integración de servicios, a través del modelo de bajo acoplamiento de los WS. Esto proporciona un entorno extensible y estándar basado en XML para la integración de servicios, a través de la utilización de SOAP/WSDL, lo cual permite la fácil integración de SIP en las transacciones empresariales. En este enfoque, un terminal WSIP puede integrarse remotamente con cualquier aplicación autorizada, y de una manera distribuida, a través del método estándar de los WS. No hay necesidad de cambiar el protocolo de señalización SIP, y no se requiere que una aplicación posea un UA SIP con el fin de transmitir el contenido y el control del servicio. La ubiquidad del entorno de integración de servicios se logra a través del enlace de SOAP, con HTTP, y de la interfaz común de descripción de servicios, WSDL.

En la Figura A19 se muestra el diagrama de comunicación de WSIP. Dos terminales WSIP se pueden comunicar directamente utilizando la señalización SIP, como se muestra en el diagrama con la línea continua. Además, dos terminales WSIP también se pueden comunicar directamente a través de WS utilizando SOAP. De manera adicional, los terminales WSIP se pueden comunicar con aplicaciones remotas que integren a SIP en los servicios, y de forma directa a través de un canal de WS (representado con la línea punteada).

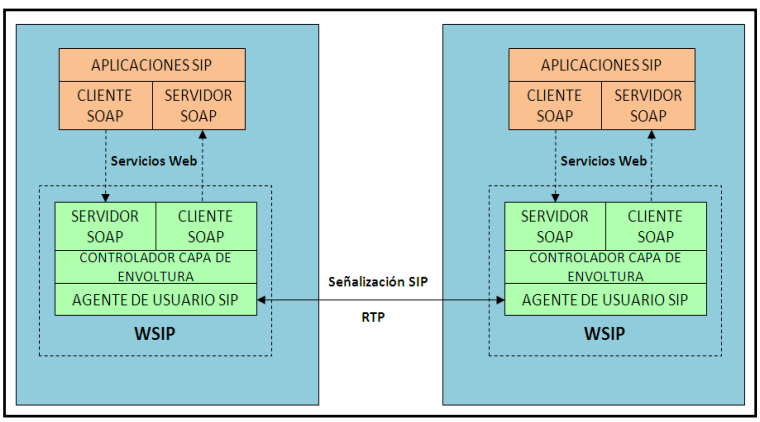


Figura A19. Arquitectura de WSIP

A.1.6. WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP

La ubicuidad de Internet y de la red IP ha hecho cada vez más importante y popular los servicios de comunicaciones sobre IP. SIP es uno de los protocolos más importantes para las comunicaciones VoIP. A pesar del éxito de SIP como el protocolo principal para VoIP, SIP no está basado ni en XML ni en servicios Web. El se basa en un conjunto de operaciones especiales de SIP para el establecimiento de sesiones y configuración de llamadas. Para soportar servicios de comunicaciones nuevos, SIP se encuentra en constante expansión mediante la adición de infaestructuras y operaciones, las cuales no se encuentran en su alcance original. En consecuencia, la infraestructura de extensibilidad actual de SIP se ha convertido en un problema. El esquema de codificación en texto simple de los encabezados SIP, y los encabezados tipo SMTP que utiliza SIP, son tanto una fortaleza como una limitación.

Un WS es una tendencia que ha ido apareciendo en la industria para la provisión de servicios distribuidos sobre IP. Este hace uso de un mensaje estructurado XML (SOAP) y de WSDL para acceder, controlar e integrar varios servicios de manera remota y para transacciones complejas. El uso de SOAP/WSDL en los WS introduce una SOA, débilmente acoplada, extensible, y muy adecuada a la infraestructura de Internet. Las nuevas especificaciones desarrolladas de los WS tales como WS-Addressing, WS-Session, WS-ReliableMessaging, WS-Eventing, etc., hacen que se avance de los WS convencionales, de petición/respuesta, a la integración de servicios de dos caminos, es decir una interacción de servicios Web full duplex sobre IP.

La comunicación convergente de multimedia y voz sobre IP es un área de investigación activa debido a su alta demanda y a su naturaleza quebrantadora. Se han propuesto muchos enfoques para fomentar el poder de las comunicaciones basadas en IP. Uno de esos enfoques es WSIP (Servicios Web + SIP), el cual es una solución de doble pila que utiliza los WS para separar la integración de servicios de la señalización y de la transmisión multimedia. Este utiliza un protocolo de la capa de aplicación superpuesto, como por ejemplo HTTP, y SIP, para la comunicación y para las extensiones del servicio. Sin embargo, no siempre puede ser viable tener la pila de SIP en una plataforma de WS, o incluso puede que no sea deseable mantener dos pilas de protocolos de comunicaciones.

En consecuencia a lo anterior en (Chou, y otros, 2006) se introduce y define un protocolo de WS denominado Protocolo de Inicio de Servicios Web (Web Service Initiation Protocol, WIP) para comunicaciones multimedia y de voz sobre IP. Este enfoque se encuentra basado completamente en WS, y en una única pila de WS para la señalización de los servicios de comunicación. Específicamente, se basa en algunos avances recientes en las tecnologías para comunicaciones de los WS, como el protocolo genérico de WS para el establecimiento de sesión: WS-Session (ECMMA-366).

WIP consiste en un conjunto de operaciones de WS para el inicio y el establecimiento de servicios de comunicaciones convergentes (por ejemplo multimedia, mensajería instantánea, voz, etc.) sobre IP. WIP hereda de SIP el principio de separación de la transmisión de multimedia de la de señalización; sin embargo se basa solamente en una pila de WS para proporcionar un protocolo de señalización de comunicaciones completamente equipado,

dejando por fuera de acción a la necesidad de convergencia con aquellas aplicaciones y servicios, basadas en la señalización SIP del dominio de las Telecomunicaciones.

WIP abre un nuevo paradigma de WS basados en comunicaciones VoIP, el cual puede extenderse y se puede integrar fácilmente en soluciones extremo a extremo basadas en SOA. El enfoque genérico de WS que se utiliza en WIP supera muchas de las limitaciones que de otra manera serían difíciles de alcanzar con los métodos de comunicación que no se basan en WS y que se utilizan hoy en día, pero es la misma fortaleza de este enfoque la que limita su campo de acción ya que desconoce la amplia adopción que se ha hecho de SIP para el soporte a la señalización en las NGN.

A.1.7. Invocación de Servicios Web por medio de la señalización SIP

En la alternativa presentada en (Fornies, y otros, 2007) se implementó la invocación de WS por medio de la señalización SIP, donde esta última es aplicada para resolver los problemas de disponibilidad e identificación de los WS.

Actualmente es posible utilizar un dispositivo móvil como un consumidor de WS, para lo cual es necesario crear un cliente en la aplicación del terminal y consumir el método correspondiente del WS para obtener los datos deseados. Sin embargo, cuando se consume una operación de un servicio, se deben tener en cuenta dos aspectos que pueden obstaculizar la interacción con dicho servicio. Por un lado, el dispositivo que aloja el servicio puede no estar disponible en el momento en que se hace la comunicación. El otro problema que se puede encontrar es que la identificación del servicio, que está contenida en su descriptor, corresponda a un espacio o dominio cuyo acceso no está disponible o es desconocido (como por ejemplo, un dominio privado de red).

Hoy en día, no hay una solución técnica para el problema de la disponibilidad de los servicios, en entornos móviles y distribuidos. El protocolo SIP y los registros hacen posible la determinación de la disponibilidad de los dispositivos, pero no se ha resuelto la disponibilidad de los servicios y de las operaciones que ellos proporcionan. Por ejemplo, en una aplicación de una patente Japonesa, la JP 2004-356967, se describe la referencia entre los WS y SIP, para simular un sistema de mensajería asíncrono y en tiempo real, conectado a HTTP. El objetivo de esta patente se centra en lograr la notificación asíncrona de mensajes, utilizando el protocolo de transporte HTTP. En otras alternativas, como la patente de la US (US 2005/0778705) y la de Japón (JP 2003-350594), se describen sistemas proveedores de servicios, los cuales transfieren información entre sistemas SIP y sistemas Web. Sin embargo, ninguna de las alternativas anteriores resuelve el problema de la disponibilidad de los servicios y de las operaciones que proporcionan tales servicios.

En (Fornies, y otros, 2007), se logra el acceso a los servicios proporcionados por dispositivos móviles con acceso a redes IP, mediante la tecnología de los WS, y utilizando el protocolo SIP como señalización.

Básicamente, el dispositivo que provee el WS está identificado por una dirección o URI, la cual es una dirección SIP, y se descubre y localiza mediante el inicio de sesiones SIP. Un usuario del WS genera la petición de inicio de sesión, indicando como URI de la petición la URI SIP del

dispositivo proveedor, y describiendo la sesión mediante SDP. Con ello se describe el contexto de la interacción, para negociar el consumo del servicio; dicho contexto incluye por lo menos la URL donde se puede invocar al WS, y un Patrón de Intercambio de Mensajes (Message Exchange Pattern, MEP). El proveedor genera una respuesta para el inicio de la sesión, descrita mediante SDP, incluyendo el contexto de la interacción que fue propuesto por el cliente o modificado. De esta manera, si el proveedor del WS está registrado en aquella red SIP, el usuario cliente accede al servicio utilizando la URI SIP del dispositivo, donde está alojado el proveedor del servicio. Un dispositivo que actúa como proveedor de servicios necesita estar disponible en una dirección de red para que los clientes puedan invocar tales servicios y consumirlos. En esta alternativa se establece que la dirección o URI que identifica un dispositivo sea una URI SIP, la cual es la dirección que se utiliza en una red SIP para identificar un usuario registrado y asociarlo con una dirección de red física donde esté disponible.

De esta forma, la alternativa define un sistema el cual permite el descubrimiento, localización y el consumo de servicios alojados en un proveedor de WS, mediante el protocolo de señalización SIP y el protocolo de descripción de sesiones SDP, que definen el contexto de interacción necesario para el intercambio de los mensajes SOAP, correspondientes a la invocación de una operación específica de un WS. El contexto de la interacción es entendido como el conjunto de datos necesarios para el consumo de la operación de un servicio que pertenece a los WS del proveedor. Esta información está conformada por un conjunto de elementos que definen las características del acceso al servicio tales como: el protocolo utilizado para tener acceso a la operación, la conexión de red, la URL de destino del servicio, el servicio, la operación y el MEP. La utilización de un MEP específico determina el ciclo de vida de la sesión que se inició para el intercambio de mensajes. Para la operación se puede utilizar el mismo canal de comunicación para el mensaje de petición y el mensaje de respuesta, lo cual es una interacción síncrona; aunque también es posible que, para el mismo contexto de la operación, el cliente y el proveedor intercambien un conjunto de mensajes de manera asíncrona, o que se trate de una operación asíncrona, en la cual el cliente envía un mensaje luego de lo cual la sesión es interrumpida.

Mediante el sistema propuesto, los servicios con direcciones de red dinámicas se pueden localizar fácilmente mediante el registro SIP. Además, el control de la presencia permite conocer de antemano la disponibilidad del servicio que se desea consumir.

A.1.8. Akogrimo: hacia una Arquitectura IMS extendida

La tendencia de la fusión de las infraestructuras de Telecomunicaciones con las infraestructuras tradicionales de las IT está en curso. En el área de las Telecomunicaciones, IMS es una plataforma de servicios promisoria, pero su rango de aplicabilidad está ligado a servicios con capacidades SIP. Paralelo a este desarrollo, está la tendencia de las SOA. Por tanto, se cree que integrando SOA e IMS se ampliará el rango de aplicabilidad de IMS, virtualizando todos los recursos de IMS a través de los WS, y para ello en (Jähnert, y otros, 2007) (Villagrá y Wesner, 2007) se describe una solución para fusionar las aplicaciones basadas en SIP y SOAP.

Los autores de (Jähnert, y otros, 2007) creen firmemente que estos conceptos se necesitan con urgencia, para poder ser finalmente exitosos en el mercado. Por tanto, se requiere del

concepto de la integración SIP-SOAP para poder llevar servicios de valor añadido basados en los WS a la red móvil.

IMS está siendo adoptado entusiastamente tanto por los investigadores como por los industriales. Sin embargo, IMS apunta solamente a aplicaciones SIP. En paralelo a IMS, la tradicional comunidad de computación distribuida (que finalmente evolucionó hacia la comunidad Grid/Web Services) ha generado soluciones y conceptos, los cuales también podrían utilizarse para la paquetización de servicios. Por ejemplo la idea de Organización Virtual (Virtual Organisation, VO) o de Organización Virtual Dinámica Móvil (Mobile Dynamic Virtual Organization, MDVO). Esto hace posible la creación de estructuras recursivas, con múltiples capas de proveedores "virtuales" de servicios de valor agregado.

El protocolo de señalización clave para la VO es SOAP: un protocolo liviano basado en mensajes para el intercambio de información estructurada en un entorno descentralizado y distribuido, a través de una variedad de protocolos subyacentes. SOAP hace posible el enlace y la utilización de WS descubiertos, mediante la definición de un camino para el mensaje por medio de mensajes de enrutamiento.

Arquitectura orientada a servicios de Akogrimo

La arquitectura orientada a servicios de Akogrimo suministra una plataforma abierta de servicios que da soporte a todo el ciclo de vida de una aplicación organizacional, distribuida, y transversal. Empezando desde la Identificación de los Proveedores de Servicios necesarios, se da soporte a la articulación de la colaboración y la implementación de una Organización Virtual Operativa (Operative Virtual Organization, OpVO), incluyendo diferentes proveedores por medio de Acuerdos a Nivel del Servicio (Service Level Agreements, SLA), y la promulgación de flujos de trabajo, enriquecidos semánticamente, en estas coaliciones creadas bajo demanda. Durante esta etapa de operación, y a lo largo del tiempo, los participantes pueden estar sometidos a una evolución, o inclusive podrían ser reemplazados o removidos completamente del proceso de colaboración. Este reemplazo puede ser causado por azares del contexto, por ejemplo un participante cambia, de manera transparente, a un terminal con diferentes capacidades, o puede cambiar la disponibilidad del ancho de banda en una red de acceso móvil. Además, es un concepto único la integración en un único flujo de trabajo de los recursos virtualizados, tales como el cómputo o el almacenamiento, servicios de las aplicaciones, y también las comunicaciones multimedia, y la virtualización de los recursos de red. El último objetivo de Akogrimo es proporcionar una plataforma abierta de aprovisionamiento de servicios para los Operadores de Red y los Proveedores de Servicios.

La integración de Akogrimo e IMS producirá beneficios mutuos: beneficios para Akogrimo de IMS y sus servicios de red, y la gama de aplicaciones IMS se incrementará. Desde el punto de vista técnico, Akogrimo se construye en la parte superior de los WS. Pero los WS utilizan SOAP y no SIP, por lo que para permitir la integración de Akogrimo e IMS se necesita diseñar la interoperación entre SIP y SOAP.

Akogrimo e IMS: Interconexión de SIP y SOAP

El concepto de combinar a SIP y SOAP consiste en tratar uno de los mayores problemas: combinar la provisión de aplicaciones basadas en la Web con la noción de movilidad y ubiquidad de los usuarios y servicios.

Existen diferentes enfoques para combinar a SIP (sentido de la noción de movilidad y ubiquidad) y SOAP (aplicaciones basadas en WS):

- En un primer enfoque se trató de incluir a SIP como protocolo de transporte para los mensajes SOAP (SOAP sobre SIP). De esta manera, los mensajes SOAP serían llevados en la carga útil de SIP, utilizando los mecanismos de enrutamiento y direccionamiento de SIP, de la misma forma que se está utilizando actualmente para la aplicación de Mensajería Instantánea. Este enfoque condujo a varios inconvenientes, siendo el más importante el hecho de que SIP se utiliza en lo alto de un protocolo no fiable como UDP, lo cual lo hace no apropiado para grandes mensajes, como aquellos que se necesitan en las interacciones SOAP, y también el hecho de que el mapeo de SOAP sobre SIP no está estandarizado, lo cual implica que no se pueden utilizar las actuales infraestructuras de WS.
- Un segundo enfoque consiste en que las aplicaciones de WS hagan uso del mismo enfoque que utilizan las aplicaciones tradicionales de VoIP: utilizar a SIP como protocolo de señalización para la localización del servidor y la iniciación de una sesión, y luego en esta sesión utilizar a SOAP para las invocaciones normales de WS. El principal inconveniente de este enfoque es que se necesita hacer Servicios Web SIP-aware (“sippifying applications”).

En Akogrimo se ha escogido este segundo enfoque, para la localización de los proveedores de servicios y sus servicios dinámicamente, y para el soporte del ciclo de vida en las VO.

Entonces, como los servicios se pueden mover desde un lugar hacia otro, o se pueden terminar e iniciar dinámicamente, no solamente será necesario localizar los servicios, sino también mantener una sesión con ellos, con el fin de rastrear las conexiones, desconexiones, re-conexiones, pausas, reanudaciones, de la misma forma en que se establecen las sesiones SIP.

Esto implica otro problema: los WS utilizan Referencias del Punto Final (End Point References, EPR) para hacer referencia a los servicios. Entonces la señalización SIP necesita proporcionar una traducción de las referencias genéricas utilizadas en SIP (Uniform Resource Identifier – URIs) a las EPRs. Un enfoque similar se utiliza en las aplicaciones tradicionales de VoIP: la señalización SIP se utiliza para localizar el códec, los puertos, etc., de las aplicaciones de streaming. Entonces, se puede utilizar el mismo enfoque para las sesiones de WS: las cargas útiles de SIP serán compuestas por un protocolo de descripción (GSDP – Grid Session Description Protocol) el cual incluye el EPR específico que el cliente necesita utilizar para invocar sus WS.

Para llevar servicios de valor agregado, y basados en WS, a la red móvil se requiere de un concepto para la integración de SIP-SOAP. La solución propuesta, la cual ha sido implementada en un prototipo y demostrada dentro del proyecto Akogrimo, es un fundamento promisorio.

La arquitectura de Akogrimo nunca se ha dirigido a reemplazar la infraestructura de IMS sino a mejorarla, y así se pueden proteger las inversiones que ya se ha hecho en el área de IMS y un

proveedor de servicios u operador de red tiene la flexibilidad de integrar sin problemas los mecanismos propuestos para enriquecer su plataforma de servicios.

A.1.9. FOKUS Open SOA Telco Playground, un Entorno de Telecomunicaciones basado en SOA

En septiembre de 2007 el FOKUS Open SOA Telco Playground se abrió oficialmente para encargarse de la Investigación y el Desarrollo en la capa de aplicación de IMS, de cara a los nuevos habilitadores IMS y su orquestación (con elementos como el SCIM, para la intermediación de servicios), de los principios de SOA y de las APIs de la Web 2.0. En este sentido, Fokus se encarga, entre otras cosas, de la creación, orquestación, intermediación, ejecución, y el aprovisionamiento de servicios, además de la gestión de servicios NGN basados en SOA, etc.

Hacia una SOA sobre IMS

- Una gran cantidad de trabajo en el Open IMS Playground se ha llevado a cabo en el contexto de las aplicaciones IMS, experimentado la integración de IMS y SDP.
- Hoy en día SOA se utiliza para describir una plataforma de prestación de servicios, que incluye la reutilización de componentes de servicio y la orquestación de servicios.
- Por lo tanto, en el 2007 se estableció el Open SOA Telco Playground en la parte superior del Open IMS Playground, como se observa en la Figura A20.

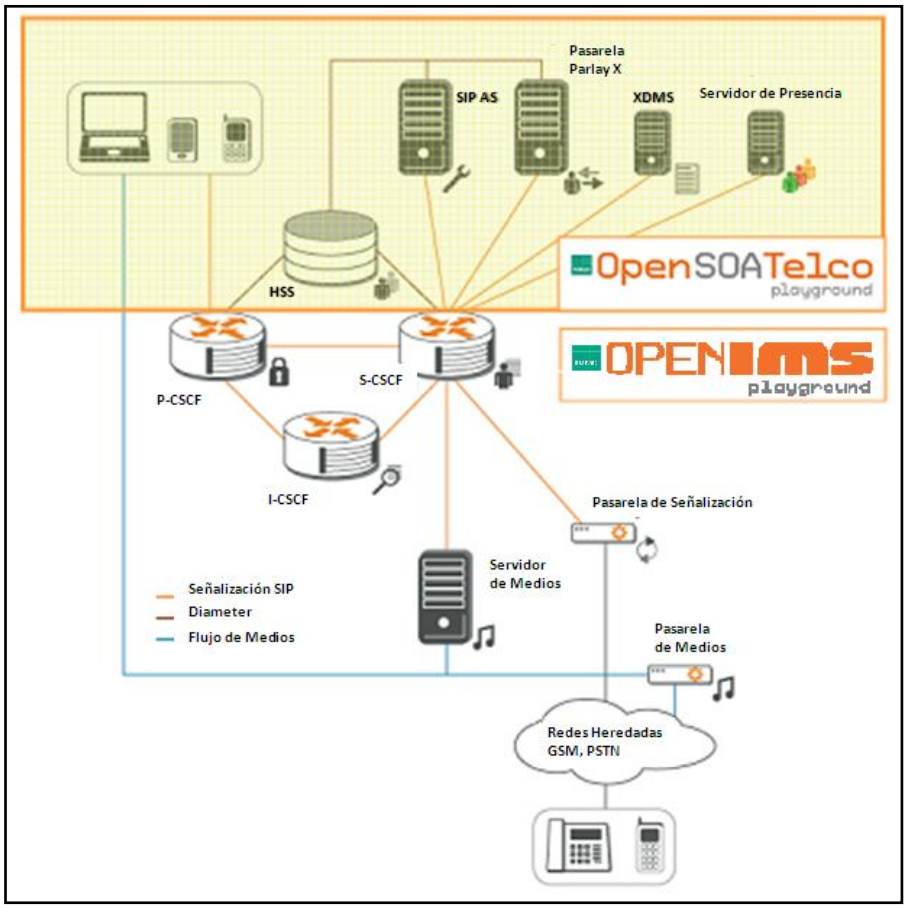


Figura A20. Open SOA Telco Playground y Open IMS Playground

Un Entorno de Telecomunicaciones basado en SOA

El trabajo que se está llevando a cabo en (Carvalho de Gouveia y Magedanz, 2008) se resume a continuación:

- Integración de los servicios de telecomunicaciones y los de la Web 2.0 bajo la utilización de los principios de SOA, centrándose en ambos sistemas finales así como en la orquestación de servicios.
- Desarrollo de intermediadores avanzados de servicios para servicios IMS (basados en SIP) y no IMS (basados en HTTP).
- Desarrollo y extensión, basada en SOA, de servidores de aplicaciones IMS.
- Desarrollo de prototipos de servicios SOA en la parte superior de IMS.
- Diseño y Desarrollo de una solución de gestión extensible para entornos NGN/IMS basados en SOA, incluyendo los sistemas de aprovisionamiento de servicios para entornos IMS basados en SOA, sistemas de monitoreo y gestión de fallas.
- Desarrollo de Infraestructuras de Comunicaciones Autónomas basadas en SOA.

La misión del Open SOA Telco Playground es apoyar a las empresas de telecomunicaciones en el diseño de su estrategia individual SOA y en la implementación de soluciones específicas, así como en el desarrollo de nuevos procesos de negocio, al tiempo que mejoran los flujos de trabajo y hacen uso de los sistemas heredados dentro de una empresa que se extiende hacia SOA. (Magedanz, y otros, 2008)

A.1.10. WIMS 2.0

En los últimos años ha habido un extraordinario avance en el mundo de las telecomunicaciones, pasando los operadores a ofrecer una gran variedad de servicios de telecomunicación que, no obstante, no han conseguido desplazar a las llamadas de voz como el servicio más rentable (si bien es cierto que el crecimiento de ingresos por este servicio se ha estancado, haciéndose necesario buscar nuevas fuentes de ingresos significativas). En paralelo con esta evolución de los servicios de telecomunicación, ha ocurrido en el mundo de Internet la revolución web 2.0, cuya fórmula de éxito se ha basado en situar al usuario en el eje central del desarrollo de servicios. En la filosofía de la Web 2.0 el usuario es el protagonista, participa en la definición de servicios y usa Internet como una plataforma de la que obtiene los servicios que desea, mezclando funcionalidades (o haciendo mashups) y contenidos de diferentes fuentes.

La iniciativa WIMS 2.0 (González, y otros, 2008), promovida por Telefónica y desarrollada técnicamente por Telefónica I+D, analiza estas tendencias y, en concordancia con los planteamientos de la iniciativa Telco 2.0 (Telco 2.0, 2009), plantea que: si los operadores de telecomunicaciones desean obtener nuevos servicios exitosos, deben abrir sus redes y ser proveedores de plataformas para servicios de Terceros, para así conseguir que lo demandado por los consumidores sea lo ofertado por el propio operador.

En (González, y otros, 2008) se analizan las posibilidades de interfuncionamiento y convergencia entre la Web 2.0 y una red IP de un operador basada en IMS, con especial énfasis en la exposición de capacidades de Telecomunicación al mundo Web 2.0, a través de un

Servidor de Aplicaciones IMS (ASIMS) que implementa la capa de adaptación entre ambos mundos.

Motivaciones y estrategias

El subsistema IMS, visto como eje central de las redes IP de los operadores de telecomunicación del futuro, brinda nuevas formas de satisfacer las necesidades de los usuarios, dividiendo el mercado, reduciendo el Time-To-Market e incrementando la creatividad para producir nuevos servicios. WIMS 2.0 considera que IMS, junto con las capacidades de telecomunicación de los operadores, conforma una plataforma apropiada para llevar a cabo el principal objetivo buscado por la iniciativa WIMS 2.0: conseguir la convergencia entre el mundo de las telecomunicaciones y el mundo de Internet, situando al usuario en el centro y disponiendo los medios para crear nuevos servicios ricos y variados, que el usuario define a su gusto e incluso colabora en su creación.

Existen dos vertientes complementarias para llegar a la convergencia deseada: ofrecer capacidades del operador a la web 2.0, y utilizar la web 2.0 para enriquecer los servicios del operador.

- **Ofrecer capacidades del operador a la web 2.0**

Con esta estrategia se pretende ofrecer funcionalidades IMS de la red de telecomunicaciones hacia la web 2.0. La iniciativa WIMS 2.0 explora conceptos vanguardistas para la convergencia con la web 2.0, centrándose a medio y largo plazo en las capacidades IMS. No obstante, la exposición de capacidades de telecomunicación legadas -no basadas en IMS- es perfectamente viable y, de hecho, a día de hoy más fácilmente demostrable y comercializable, debido precisamente a que IMS representa el futuro de las redes de telecomunicaciones.

De acuerdo con esta aproximación, el servicio final podría ser ofrecido por un Tercero situado en la web 2.0, aportando el operador un valor añadido y manteniéndose con un rol activo en la cadena de valor, más allá de ser un mero proveedor de conexión. Dentro de esta estrategia se pueden diferenciar dos líneas de actuación:

1) Proporcionar funcionalidades de servicios IMS en la web 2.0, habilitando la integración de IMS en los mashups de la web 2.0. Se han identificado dos modos de llevar a cabo esta línea de acción:
   a) Portable Service Elements (PSEs), esto es, aplicaciones IMS que pueden ser incrustadas, como los widgets web, en la web 2.0, por elección de los usuarios y dotándoles de posibilidades de configuración. Estos PSEs, ofrecidos por el operador y disfrutados por el usuario, son capaces de interactuar con los APIs abiertos del operador para habilitar así el uso de las capacidades de comunicación IMS por parte del usuario.
   b) Incorporación directa en los mashup de la web 2.0, lo que supone que el operador expone de forma controlada APIs abiertos que ofrecen capacidades de comunicación. Cualquier servicio web 2.0 podrá hacer uso de estos APIs para crear los mashups a incorporar en el servicio web 2.0. El resultado final es una integración completa de las capacidades de telecomunicación en el servicio web 2.0 ofrecido por un Tercero,

ampliando así extraordinariamente las posibilidades de adaptación al usuario y pasando el operador a ser un proveedor relevante en el mundo de Internet.

2) Publicación, habilitada por IMS, de contenido generado por el usuario: La web es el principal repositorio de contenido y, cada vez más, los usuarios son los principales creadores de contenidos. Esta línea persigue que los usuarios publiquen contenido en la web 2.0 mediante las capacidades de transmisión de medios IMS disponibles en sus terminales. Tal funcionalidad será de vital importancia, ya que se espera que los terminales de las telecomunicaciones sean una de las principales fuentes de contenido en un futuro cercano.

- **Utilizar la web 2.0 para enriquecer los servicios del operador**

Esta estrategia complementa a la anterior y plantea los medios técnicos para que el operador de telecomunicaciones explote las posibilidades de la web 2.0 en sus propios servicios. De nuevo, se consideran dos líneas de actuación:

1) Incorporar contenidos y eventos de la web 2.0 en los servicios de telecomunicaciones. Debido a que la mayoría del contenido se aloja en la web 2.0, se persiguen nuevos mecanismos para obtener este contenido de la web 2.0 y transmitirlo a los usuarios IMS a través de las capacidades de telecomunicación IMS. Así mismo, se considera también la incorporación de eventos web 2.0, entendidos en un sentido amplio. Por ejemplo la presencia y las actividades realizadas por los usuarios en las redes sociales y otros sitios web 2.0, la incorporación de la información de contacto, noticias, sincronización de calendarios, etc.
2) Aplicaciones IMS on-line, de forma que las aplicaciones de telecomunicaciones no se sitúen en el terminal sino en la red en la forma de aplicaciones web, por lo que aportarían ubicuidad y un desarrollo y despliegue de servicios más simplificado y versátil.

Ejecución práctica de la exposición de capacidades

La principal aproximación seguida para llevar a la práctica los conceptos de WIMS 2.0 ha sido la exposición de capacidades del operador hacia la web 2.0. A continuación se exponen las bases tecnológicas seguidas y los casos de uso desarrollados.

- **Bases técnicas**

La exposición de capacidades hacia la web 2.0 se fundamenta en el concepto de APIs web abiertas que faciliten la interacción entre el mundo web, basado en el protocolo HTTP, y las capacidades IMS, basadas en SIP y el Protocolo de Configuración del Acceso XML (XML Configuration Access Protocol, XCAP). Para tal fin, se pueden seguir dos filosofías basadas, respectivamente, en la orientación a procedimientos (Llamada a Procedimiento Remoto (Remote Procedure Call, RPC) y en la orientación a recursos, mediante el API para la Transferencia de Estado Representacional (Representational State Transfer, REST) (Costello, 2002).

El mejor exponente del modelo RPC es el protocolo SOAP para los WS, ampliamente adoptado en el mundo de las telecomunicaciones y en el que se basan, entre otros, los WS de ParlayX.

Para el segundo modelo, no existe un protocolo concreto. REST es una filosofía general basada en principios como la identificación de recursos mediante URIs, el uso de los métodos HTTP para el manejo de estos recursos, o el paradigma cliente/servidor sin estado.

Mientras que SOAP está más extendido y lleva más tiempo utilizándose en el mundo de las telecomunicaciones, las implementaciones basadas en REST están aumentando vertiginosamente en la web 2.0 y son ampliamente empleadas en la realización flexible y desacoplada de mashups. Este aumento de REST se debe a la simplicidad de sus planteamientos, su versatilidad y la ligereza de las transacciones, al reutilizar la semántica de HTTP, protocolo clave en la web. Por estos motivos, la iniciativa WIMS 2.0 aboga por la exposición hacia la web 2.0 de capacidades del operador mediante APIs de REST. No obstante, dependiendo de factores como la complejidad del caso de uso o el escenario de aplicación, desde la iniciativa WIMS 2.0 no se descarta el uso de APIs SOAP, como por ejemplo APIs de ParlayX.

Otro aspecto a tener en cuenta es el formato de datos a utilizar en las APIs para la exposición de IMS. Para SOAP este formato viene fijado por el propio protocolo mientras que para los interfaces REST se considera apropiado ofrecer diferentes alternativas como XML plano, RSS/Atom y JSON. Así mismo, para facilitar el uso de las APIs se considera interesante ofrecer librerías cliente en diferentes lenguajes de programación como Java, PHP, Python o Ruby, así como ofrecer libertad del lado cliente para elegir el formato de datos en el que recibirá la eventual respuesta.

Por último, es evidente que, al pretenderse dar acceso a los servicios de telecomunicaciones, a través de mecanismos web, se deberá fijar un sistema de autenticación/autorización que ofrezca garantías acerca de la identidad y el respeto de la privacidad en el lado web 2.0, máxime cuando se ofrezca la posibilidad a Terceros de acceder a las capacidades del operador de telecomunicaciones en nombre de los usuarios finales, para así ofertar sus propios servicios a los clientes.

- **Casos de uso que validan el concepto WIMS 2.0**

Como parte de la iniciativa WIMS 2.0 y, tras el planteamiento de las diferentes líneas de convergencia, se ha definido un modelo de referencia (Figura A21) de una plataforma de servicios WIMS 2.0.

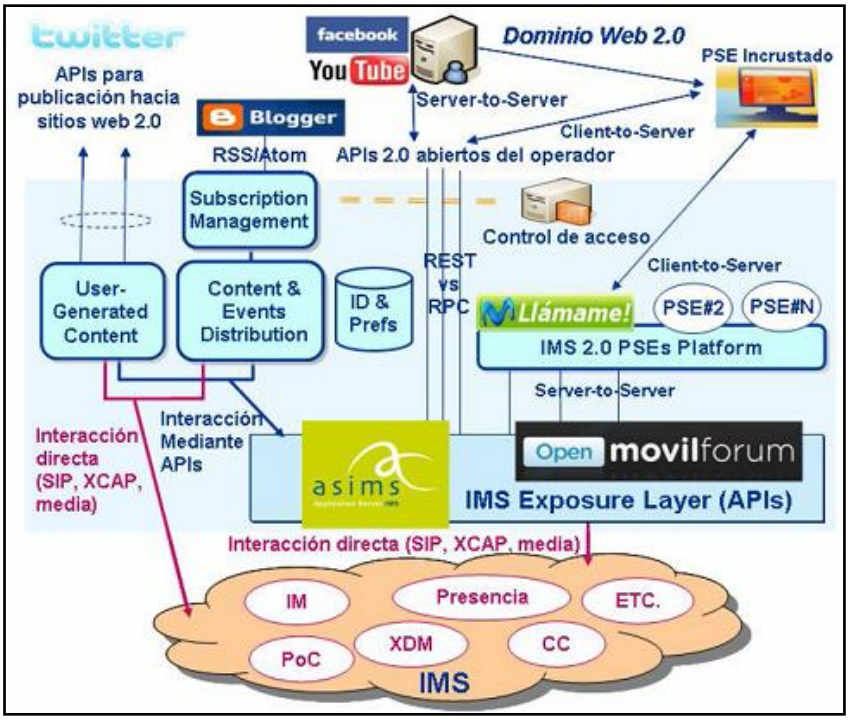


Figura A21. Modelo de referencia de la Plataforma de Servicios WIMS 2.0

Como se ha comentado, la línea de convergencia seguida principalmente en los casos de uso implementados es la de exposición de capacidades IMS mediante APIs abiertos, que son utilizados bien para proporcionar funcionalidades de servicios IMS en la web2.0 (mediante mashup directo o a través de PSEs) o para publicar mediante IMS contenido generado por el usuario en la web 2.0. Como se aprecia en la Figura A22, la exposición de los APIs hacia la web 2.0 es realizada por una entidad que la iniciativa WIMS 2.0 ha nombrado como "IMS Exposure Layer" o "Capa de Exposición de IMS". Básicamente, esta entidad realiza labores de Pasarela e interactúa (mediante APIs o directamente empleando SIP/XCAP) con las capacidades del operador por un lado y con la web por el otro. En los diferentes casos de uso, esta entidad se ha implementado, bien a través de los APIs del Open movilforum, o bien mediante un Servidor de Aplicaciones IMS (ASIMS).

Los siguientes casos de uso, que han sido desarrollados e implementados como parte del trabajo llevado a cabo desde la iniciativa WIMS 2.0, representan implementaciones de las entidades ilustradas en el modelo de referencia y pueden ser demostrados como resultado práctico de los conceptos WIMS 2.0 y de la potencialidad del ASIMS como plataforma para aplicar estos conceptos:

1. *Click2Talk para Blogger*: Permite al dueño de un blog, de Blogger, personalizarlo añadiendo un widget, llamado "Botón Llámame", que permite a los visitantes del blog establecer una llamada entre el dueño del blog y el teléfono indicado por el visitante de dicho blog.
2. *Click2Talk para PYME*: Este caso de uso es conceptualmente similar al anterior, si bien está orientado a otro tipo de usuarios que también tienen su papel en la web 2.0: las PYME. Se proporciona a las PYME una herramienta que permite insertar en sus webs públicas un servicio de comunicación enriquecido que permite a los clientes, visitantes online,

establecer una llamada entre el teléfono del cliente y el departamento de atención al cliente de la PYME, proporcionando, además, información de disponibilidad de las líneas implicadas. Este caso de uso, así como el anterior, ejemplifica los PSEs (widgets) del operador, que a través de APIs hacen uso de las capacidades expuestas por la Capa de Exposición de IMS.

3. *Publicación de nota personal en Twitter*: Permite al usuario publicar de manera automática en su microblog de Twitter una 'Nota Personal', que será publicada como parte de su información de presencia IMS. Por lo tanto, se sincroniza el mensaje personal de presencia IMS con el microblog del servicio web 2.0 Twitter. Este caso de uso ejemplifica las nuevas posibilidades en lo referente a la generación de contenido por parte del usuario de telecomunicaciones.
4. *Localízanos para Facebook*: Aplicación gadget para Facebook que permite representar gráficamente la localización de los miembros del grupo Facebook del usuario e interactuar con ellos vía SMS y MMS. Al igual que los dos primeros casos de uso, esta aplicación ejemplifica un PSE (widget) del operador que hace uso de las capacidades de telecomunicaciones, a través de la Capa de Exposición de IMS.
5. *RSS-MMS*: Permite al usuario suscribirse a un canal RSS/Atom y recibir los feeds vía MMS. Para canales de Blogger, el usuario puede dejar comentarios a la nueva entrada en el blog mediante una llamada de teléfono. Finalmente, este caso de uso ejemplifica las posibilidades de incorporación de contenidos y eventos web 2.0 en los servicios del operador.

En estos casos de uso se representan escenarios en los que usuarios de la web 2.0, generalmente cualquier usuario individual de Internet, puede obtener un servicio añadido de su experiencia de la web 2.0, con la particularidad de que ese servicio añadido se sustenta en el uso de una capacidad procedente del mundo de telecomunicaciones. Si bien los casos de uso se han desarrollado para un escenario en el que el operador ofrece directamente el servicio basado en una de sus propias capacidades de telecomunicaciones, la estrategia de exposición de capacidades permite escenarios en los que un Tercero aprovecha esas capacidades para crear él mismo el servicio.

Con estos casos de uso se realiza una implementación de las entidades clave que conforman el modelo de referencia de la plataforma de servicios WIMS 2.0, tal y como se detalla a continuación:

- En los casos de uso 1, 2, y 3 se han desarrollado APIs REST y ParlayX que exponen la capacidad de telecomunicaciones apropiada hacia la web 2.0, en concreto: un API para el establecimiento de llamadas controlado desde un Third Party para los casos de uso 1 y 2 y una capacidad para el manejo de información de presencia IMS (obtenida mediante suscripción o consulta puntual) para los tres casos de uso. Estos APIs han sido implementados en el ASIMS, que además hace las funciones de Pasarela hacia el mundo de telecomunicaciones, para llevar a cabo las operaciones requeridas. En los casos de uso 4 y 5 se ha reflejado cómo los principios de exposición de APIs son generales y pueden ser aplicados también a capacidades legadas no-IMS, puesto que los APIs de los que se sirven estos servicios son ofrecidos por Open movilforum y utilizan capacidades legadas como el envío de SMS y MMS.

Completando la validación que los casos de uso hacen del modelo de referencia presentado, se identifica cómo la publicación de información de presencia en Twitter en el caso de uso 3 y la publicación de un comentario vocal en un blog en el caso de uso 5 representan una implementación del elemento "User Generated Content enabler" o "Habilitador de Contenido Generado por el Usuario". Por su parte, el "IMS 2.0 PSE Platform" aloja y sirve, para que sean incorporados en la web 2.0, los widgets desarrollados en los casos de uso 1,2 y 4. A su vez, el elemento "Subscription Management" o "Gestión de la Subscripción" es la entidad que se suscribe a los canales de información, por lo tanto se identifica con la suscripción a canales RSS/Atom, desarrollada como parte del caso de uso 5. Finalmente, también como parte del caso de uso 5, se desarrolla una entidad para el envío de MMS, que se correspondería con la entidad "Contents & Events Distribution" o "Contenidos y Distribución de Eventos", cuya función, de una forma más general, sería la del envío de medios y eventos desde la web 2.0 hacia IMS.

ASIMS: Un Servidor de Aplicaciones IMS y Expositor de Capacidades

A continuación se describe cómo un Servidor de Aplicaciones IMS (ASIMS) es capaz de ofrecer las funcionalidades de exposición requeridas por la Capa de Exposición de IMS, que, como se ha comentado, es clave en el modelo de referencia WIMS 2.0. El ASIMS, además de realizar la exposición, actúa como Pasarela e interactúa con las capacidades del operador, siendo así una implementación completa de la Capa de Exposición de IMS, tal y como se observa en la Figura A22.

- **El papel de un Servidor de Aplicaciones en una red IMS**

Uno de los principales cometidos de una red IMS es el encaminamiento de los mensajes SIP entre los participantes en una comunicación. Dentro de los diferentes elementos de la arquitectura de una red IMS que intervienen en este proceso, cabe destacar el papel de los elementos que conforman la Función de Control de la Sesión de Llamada (Call Session Control Function, CSCF), y en especial el del S-CSCF (Serving CSCF), entre cuyas funciones se encuentra la de mantener el estado de registro de los usuarios y de las sesiones IMS entre los mismos. La interfaz ISC es el punto de comunicación entre el S-CSCF y los Servidores de Aplicaciones, que se encargan de realizar tareas complejas manipulando los mensajes SIP, ofreciendo así servicios de comunicación avanzados.

Así pues, un Servidor de Aplicaciones (Applications Server, AS) es, por un lado, un elemento de la arquitectura de red de IMS con capacidad para intercambiar mensajes SIP con el S-CSCF, y por otro, un contenedor de servicios encargado de mantener el ciclo de vida de éstos y de proporcionarles los recursos de red necesarios para llevar a cabo sus tareas.

En este contexto, Telefónica I+D ha desarrollado un AS, conocido como ASIMS, que está alineado con el estándar IMS y que proporciona una amplia gama de recursos de red (conocidos como capacidades) a los servicios, tal y cómo se escenifica en la Figura A22. Estas capacidades, detalladas a continuación, abstraen a los servicios de los detalles de la red y de los habilitadores sobre los que se sustentan.

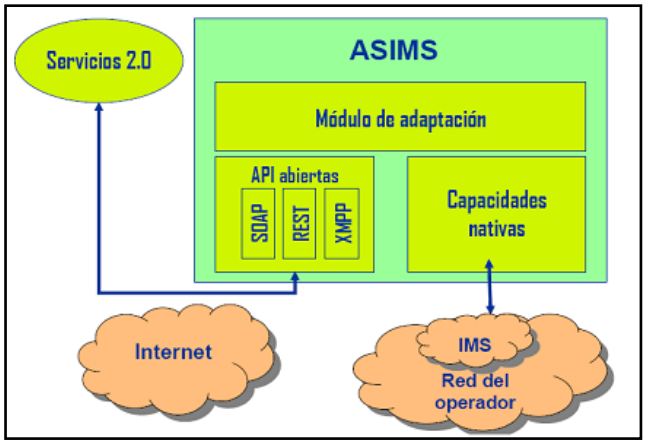


Figura A22. Modelo Arquitectura del ASIMS como expositor de capacidades (equivalente a la entidad correspondiente a la Capa de Exposición de IMS, del modelo de referencia)

- **Las capacidades de comunicación del ASIMS**

Las siguientes capacidades del ASIMS son candidatas para la construcción de APIs en el contexto de la Capa de Exposición de IMS, de WIMS 2.0:

- *Capacidad de Control de llamada*. Permite gestionar la señalización relativa al control de sesiones multimedia. Puede actuar en los roles de iniciador, terminador, proxy y B2BUA. Soporta precondiciones y simulación de servicios suplementarios.
- *Capacidad de Presencia*. Gestiona la comunicación con un Servidor de Presencia, actuando en los roles de observador y asistente. También puede actuar como Servidor de Presencia y como Agente de Red para la Presencia (Presence Network Agent, PNA).
- *Capacidad de Registro*. Permite a los servicios actuar como el Servidor Registrar (para escenarios de Registro de Terceras Partes y con suscripción al paquete de eventos de registro). También permite a los servicios registrarse ante un Servidor Registrar, implementando autenticación MD5.
- *Capacidad de Mensajería Instantánea*. Permite enviar y recibir mensajería instantánea según el modelo *Pager* descrito en el estándar OMA SIMPLE v1.0. Soporta los formatos "`text/plain`" y "`Message/CPIM`", así como las notificaciones IMDN, los mensajes ad-hoc para grupos y los mensajes de sistema.
- *Capacidad de servidor HTTP*. Permite a los servicios recibir peticiones HTTP con soporte de cabeceras y contenidos `application/x-www-form-urlencoded` y `mime/multipart`.
- *Capacidades de control de medios (MRFC/MRFP)*. Permiten a los servicios negociar sesiones con entidades como la Función de Recursos Multimedia (Media Resource Function, MRF) y con servidores de multiconferencia.

- **Apertura de las capacidades**

Una vez que se dispone de un AS que ofrece las capacidades de comunicación anteriormente indicadas, el siguiente paso para la apertura de capacidades consiste en dotar al ASIMS de las interfaces que se van a exponer, y que los servicios van a ver como la Capa de Exposición de

IMS. Esta ha sido la principal labor realizada sobre el ASIMS para llevar a cabo las pruebas de concepto detalladas anteriormente.

La apertura de las capacidades no es otra cosa que proporcionar a través de un API la totalidad o un subconjunto de las funcionalidades nativas que ofrecen las capacidades, realizando las adaptaciones necesarias. Estas adaptaciones pueden consistir en una mera traducción de primitivas, parámetros y opciones hacia las capacidades nativas, o pueden implementar lógicas más o menos complejas que permitan ofrecer un API, que en realidad es una composición de capacidades nativas, elevando así el nivel de abstracción. Como ejemplo de esto último, cabe destacar el modelo de "Third Party Call" o "Llamada de Terceros", que ofrece una API para poner en comunicación a dos usuarios, lo que se traduce en la composición de dos instancias de la capacidad de Control de Llamada, que inician sesiones multimedia hacia los destinatarios. Este modelo es la base de los casos de uso 1 y 2.

Otro aspecto clave de la apertura de capacidades se centra en la naturaleza de las APIs expuestas, ya que de ello depende el éxito de las mismas, al facilitar su uso. Para ello conviene elegir cuidadosamente las tecnologías en las que basar estas APIs, como por ejemplo REST, SOAP, XMPP, etc. Cabe destacar que la elección de estas tecnologías no sólo impone un modelo de invocación remota, sino que también condiciona la manera en que se exponen las APIs de comunicación.

A.1.11. Definición de la Arquitectura de una Plataforma de Prestación de Servicios, utilizando conceptos genéricos de IMS y SOA

Actualmente, la red de Telecomunicaciones está evolucionando hacia una Plataforma de Prestación de Servicios (Service Delivery Platform, SDP), que maneje la convergencia de las Telecomunicaciones e Internet/IT. Sin embargo, las SDP no tienen una arquitectura estandarizada. Es por ello, que en (Rolan y Hu, 2008) se define la arquitectura de una SDP basada en conceptos genéricos extraídos de IMS (Telecomunicaciones) y de SOA (Internet), ya que por sí solas, tanto IMS como SOA, proveen arquitecturas limitadas de plataformas de servicios que dependen de las tecnologías subyacentes.

IMS no define una arquitectura completa para una plataforma de servicios. Mejor que ello propone una capa de servicios limitada que está constituida, entre otros, por AS SIP, que alojan este tipo de aplicaciones. Estas últimas se comunican con las entidades funcionales de la red utilizando mensajes SIP. La capa de servicios de IMS realiza la abstracción de las entidades funcionales y proporciona un punto de referencia basado en SIP, en la cual se despliegan diversas plataformas de servicios basadas en estándares. Entre ellas se incluyen la de la Red Inteligente (Intelligent Network, IN), Parlay (Parlay, 2009), Parlay X (Parlay X, 2009), SCIM y plataformas de servicios basadas en SIP. De esta forma, las plataformas de servicios utilizan la capa de servicios para acceder a las funciones de red subyacentes.

La arquitectura de la plataforma de servicios de IMS es la integración de la capa de servicios y de las plataformas de servicios. Por tanto, para utilizar dicha arquitectura en la definición de la arquitectura de la SDP se extrajeron sus conceptos genéricos. Estos conceptos se estructuran dentro de una arquitectura generalizada de la plataforma de servicios IMS, y se muestra en la Figura A23.

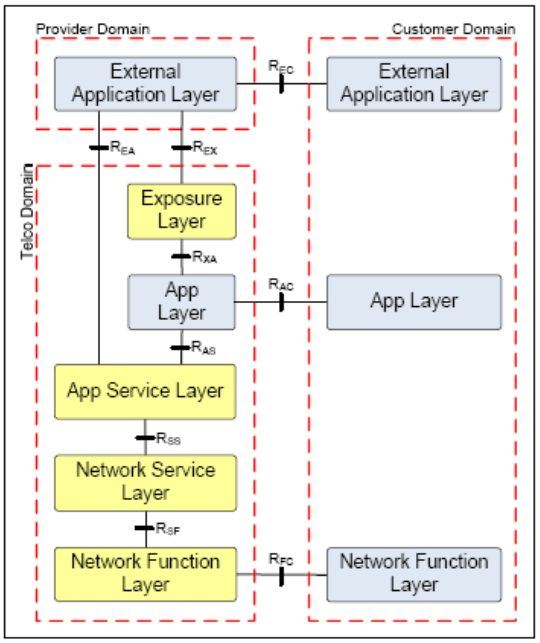


Figura A23. Arquitectura Generalizada de la Plataforma de Servicios IMS

- Las capas, en general, estructuran las abstracciones de los servicios y las funciones de red.
- Las capas de servicios abstraen las capas funcionales y pueden ser accedidas por aplicaciones externas.
- Los dominios distribuyen las capas y las abstracciones a través de las redes de Telecomunicaciones/Internet/IT. Los dominios implican la comunicación distribuida de las abstracciones a través de las capas.
- Los puntos de referencia promueven una comunicación estandarizada entre abstracciones, capas y dominios. Por ejemplo, REA y REX muestran la comunicación entre las infraestructuras de Telecomunicaciones/IT. El RSF, también, define un único punto para la integración entre servicios y funciones. Por medio de la descomposición de los puntos de referencia se descubren los detalles adicionales.

Por otra parte, para emplear a SOA en la definición de la arquitectura de la SDP, se utilizó su definición de neutralidad tecnológica, que es la SOA Genérica (Generic SOA, GSOA). La GSOA se utilizó para definir arquitecturas de plataformas de servicios independientes de los detalles de las tecnologías, la distribución y la implementación. En este sentido, la SOA basada en los WS es una implementación de GSOA. La representación de una GSOA extendida se muestra en la Figura A24. La GSOA encapsula los siguientes conceptos:

- Servicios abstractos que simplifican el acceso a las capacidades y recursos de la red.
- Las interfaces de los servicios hacen posible la accesibilidad a los servicios por parte de diversas implementaciones de aplicaciones.
- El mediador de neutralidad tecnológica oculta los detalles de implementación, distribución y de tecnología.

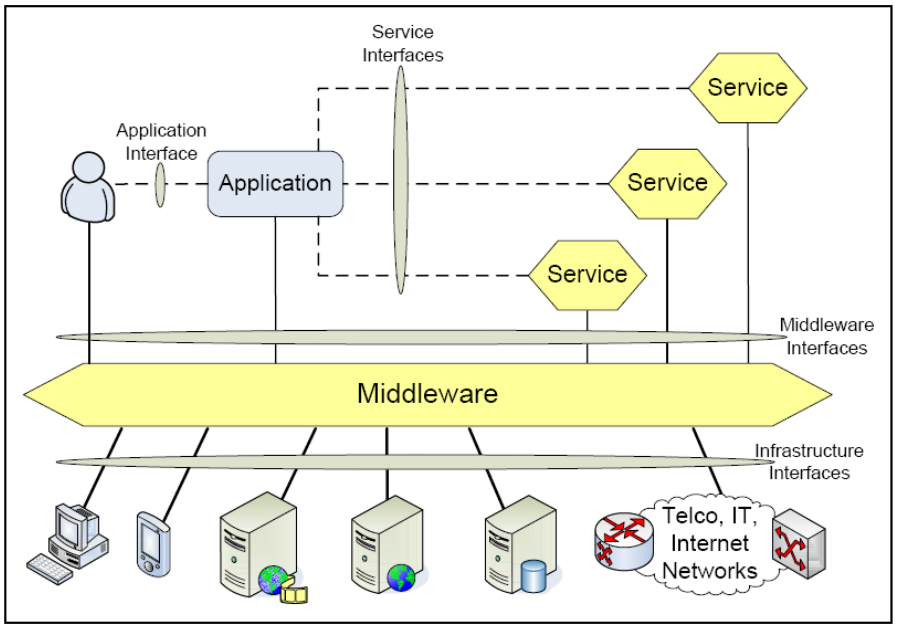


Figura A24. Representación de una GSOA Extendida

Definición de la arquitectura de la SDP

Como resultado de lo anterior, la arquitectura de la SDP integra la arquitectura de la plataforma de servicios de IMS y GSOA. La arquitectura de la SDP se muestra en la Figura A25.

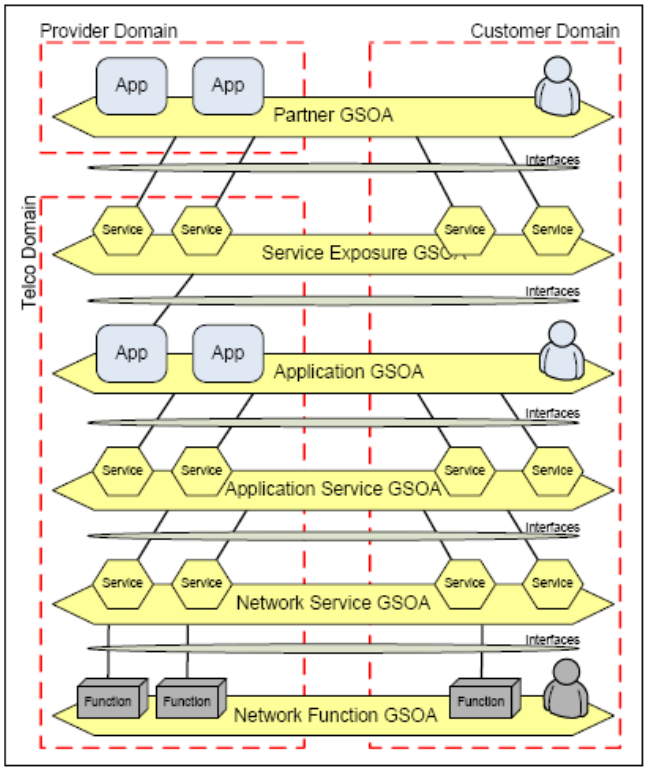


Figura A25. Arquitectura de la SDP

En la Figura, los múltiples GSOA descomponen los puntos de referencia para proporcionar detalles adicionales a la arquitectura de la plataforma de servicios. También, cada capa está

representada utilizando GSOA. Es decir, los puntos de referencia REA, REX, RXA, RAS, RSS y RSF se descompusieron en GSOA.

Dado a que las capas están estructuradas jerárquicamente, también están estratificadas sus GSOA asociadas y esto simplifica el acceso entre ellas. Esto muestra a las aplicaciones GSOA, los servicios y el mediador abstrayendo otros GSOA.

Las GSOA también abstraen las capas de dominios distribuidas. En este sentido, las GSOA descomponen los puntos de referencia REC, RAC y RFC.

A.1.12. Modelo de Comunicaciones basado en SIP para Servicios Web en Tiempo Real

Los proveedores de servicios de telecomunicaciones están buscando nuevos paradigmas para la creación y ejecución de servicios para reducir los plazos de comercialización de los mismos, y también están buscando cómo crear nuevas aplicaciones, orientadas por el mercado, para la composición del conjunto de servicios existentes. Esta ha sido una de las orientaciones del esfuerzo de los operadores de telecomunicaciones en los últimos años. Sin embargo, los actuales servicios de telecomunicaciones se prestan a través de redes separadas, y de servicios específicos, integrados verticalmente. Hoy en día, las plataformas de prestación de servicios para los servicios de valor agregado han evolucionado desde la tradicional red inteligente, hacia las recientes plataformas basadas en los WS. En estas condiciones, el foco del problema se ha vuelto el cómo proveer WS para comunicaciones en tiempo real.

Los WS permiten a los operadores de redes acelerar la adopción de los conceptos software de las IT, como SOA, en el dominio de las telecomunicaciones, lo que puede reducir el tiempo de salida de servicios de valor añadido al mercado. En el dominio de la computación, la SOA es un concepto orientado a los negocios que está basado en un estilo de desarrollo que utiliza servicios y componentes, débilmente acoplados, para dar soporte a los requerimientos de los procesos de negocios y de los usuarios. Inspirados en el éxito de las tecnologías de WS para el desarrollo y despliegue de los servicios IT, también la comunidad de Investigación y Desarrollo de las Telecomunicaciones está empezando a adaptar y desarrollar plataformas de prestación de servicios similares; con SOA los recursos de la comunicación en una red se hacen disponibles como servicios o componentes independientes, a los cuales se puede acceder y los cuales se ponen a disposición para la creación de nuevas aplicaciones o para componer servicios independientemente de la implementación de la plataforma subyacente. Los WS de telecomunicaciones se basan fuertemente en comunicaciones asíncronas para el manejo de eventos heterogéneos generados por los recursos de red, y la complejidad de dichos mecanismos, en especial, posee retos importantes para la implementación del modelo de comunicaciones para los WS en el dominio de las Telecomunicaciones.

En el dominio de las telecomunicaciones, SIP es uno de los protocolos más importantes para las comunicaciones VoIP. El crecimiento explosivo de las redes IP y la necesidad de servicios de voz sobre IP como una alternativa a la Red Telefónica Pública Conmutada acelera en gran medida la aceptación de SIP como el método principal para la VoIP de próxima generación, en el entorno de las comunicaciones convergentes. A pesar del éxito de SIP, SIP no está basado ni en XML ni en los WS, sino que se basa en un conjunto de operaciones SIP especiales para establecer sesiones de comunicaciones. En consecuencia, en (Cheng, y otros, 2008) se

presenta un panorama general del reciente desarrollo en el área de las Telecomunicaciones, centrado en las tecnologías clave que se pueden aplicar a la habilitación de WS de comunicaciones. En particular, se presenta el Modelo de Comunicaciones basado en SIP para Servicios Web en Tiempo Real (Real-Time Web Services Communication Model, RT-WSCM), diseñado a partir de máquinas de estados mejoradas para el manejo de las llamadas en el RT-WSCM.

Marco de Trabajo del RT-WSCM basado en SIP

El marco de trabajo del RT-WSCM consta de 8 sub-módulos relacionados: Módulos OAM, el Módulo de Envío de Mensajes, el Módulo de Procesamiento de los Mensajes Relacionados con la Llamada, el Módulo de Procesamiento de los Mensajes no Relacionados con la Llamada, el Módulo de Control del Servidor de Medios, Módulo de Procesos OAM, Módulo Adaptador de la Pila de SIP y la interfaz de WS. El marco de trabajo del RTWSCM basado en SIP se muestra en la Figura A26.

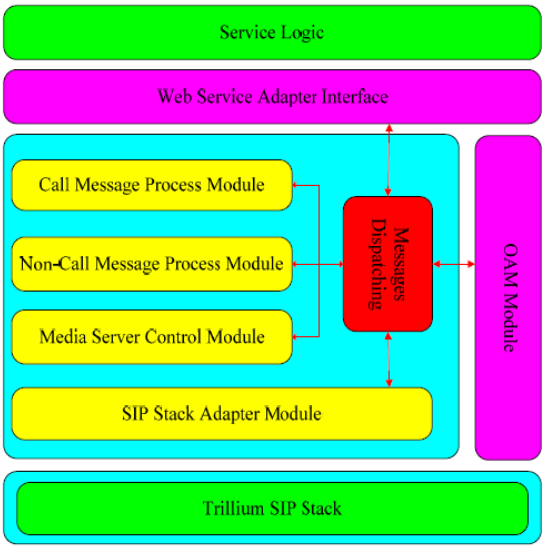


Figura A26. Modelo de comunicaciones basado en SIP para los WS

De la Figura A26 se pueden ver los módulos principales del mecanismo de implementación del modelo de comunicaciones en tiempo real, basado en SIP, para los WS, en el dominio de las Telecomunicaciones. Para cada módulo del RT-WSCM, se presentarán más detalles a continuación.

1. ***Web Service Adapter Module***, o Módulo Adaptador de WS, es el responsable de procesar los mensajes SOAP, codificarlos y decodificarlos. Cuando recibe mensajes SOAP del cliente, se encarga de transformarlos en invocaciones de WS con los protocolos correspondientes, y también es responsable de retornar el resultado con el formato correcto para el cliente que hizo la petición. También se han definido algunos servicios públicos que participan en este proceso, tales como estructuras de datos públicas, y manejo público de excepciones.
2. ***Messages Dispatching Module***, o Módulo de Envío de Mensajes, es el responsable de reenviar los mensajes para que sean procesados por la interfaz interna correspondiente, además estos mensajes provienen de los módulos de la lógica de negocio, de gestión OAM,

adaptador de la pila SIP, de procesamiento de los mensajes relacionados o no con las llamadas, y de control del servidor de medios.

3. ***Call Messages Process Module***, o Módulo de Procesamiento de los Mensajes relacionados con la Llamada, es el responsable del mantenimiento del estado del control de la llamada, y está compuesto de un sub-módulo de gestión de las máquinas de estados, y una serie de sub-módulos correspondiente a las máquinas de estados de las llamadas. El sub-módulo de gestión de las máquinas de estados es el responsable de la creación y eliminación de las máquinas de estados, y también reenvía los mensajes de petición provenientes de la lógica del servicio y del módulo adaptador de la pila de SIP, para que sean procesados en la máquina de estados correspondiente. El sub-módulo de las máquinas de estados es el responsable del procesamiento de los mensajes de petición provenientes del módulo de la lógica de negocio y de la pila de SIP, además del mantenimiento del estado del control de la llamada y de la notificación del evento de la llamada, en el establecimiento de la llamada real.
4. ***Media Server Control Module***, o Módulo de Control del Servidor de Medios, es el responsable de controlar los mensajes relacionados con el servidor de medios. Se utiliza el mensaje SIP de tipo INFO para transmitir el Lenguaje de Marcado para Servidores de Medios (Media Server Markup Language, MSML), el cual es un lenguaje XML utilizado para controlar el flujo de medios y los servicios aplicados al flujo de medios dentro del servidor. El MSML permite que durante el ciclo de vida del diálogo SIP los servicios se ejecuten y se modifiquen dinámicamente por medio de un agente de control.
5. ***Non-Call Messages Process Module***, o Módulo de Procesamiento de los Mensajes no relacionados con la Llamada, es el responsable del procesamiento de los mensajes de petición que no se relacionan con la llamada, o de los mensajes SIP provenientes de la red subyacente, como por ejemplo el registro de servicios, que lleva a cabo la gestión dinámica de la localización de usuarios.
6. ***OAM Process Module***, o Módulo de Procesos OAM, el cual re-envía los mensajes al módulo de procesos OAM, para completar la estadística de los mensajes. Estos mensajes contienen, entre otros, los mensajes de configuración y de registro.
7. ***SIP Stack Adapter Module***, o Módulo Adaptador de la Pila de SIP, proporciona la interfaz para el envío de los mensajes SIP desde el módulo de entrega de mensajes, el cual es el responsable de traducir el mensaje SIP interno al formato de mensaje de la pila de SIP. También proporciona la interfaz para recibir el mensaje SIP que proviene de la pila de este protocolo, la cual es responsable por la traducción del mensaje SIP al formato del mensaje SIP interno, y posteriormente del envío de este último al módulo de entrega de mensajes para un procesamiento futuro.
8. ***Trillium SIP Stack***, o Pila Trillium de SIP, es una implementación completa de la pila de SIP, desarrollada en código C, para su uso en un dispositivo con capacidades SIP y plataformas de telecomunicaciones de próxima generación.

Diseño de las Máquinas de Estados para RT-WSCM

En el dominio de las telecomunicaciones, las transacciones con estado se basan generalmente en una sesión que maneja un contexto con estado. Sin embargo, los métodos actuales de los WS están diseñados principalmente para la integración de servicios y se basan típicamente en un patrón de interacción de petición/respuesta sin estado. Por lo tanto, se proponen y se

diseñan máquinas de estados de las llamadas para el RT-WSCM, que constan de una máquina de estados gestora principal y diversas máquinas de estados para las llamadas. El diagrama de clases, con sus relaciones, se muestra en la Figura A27.

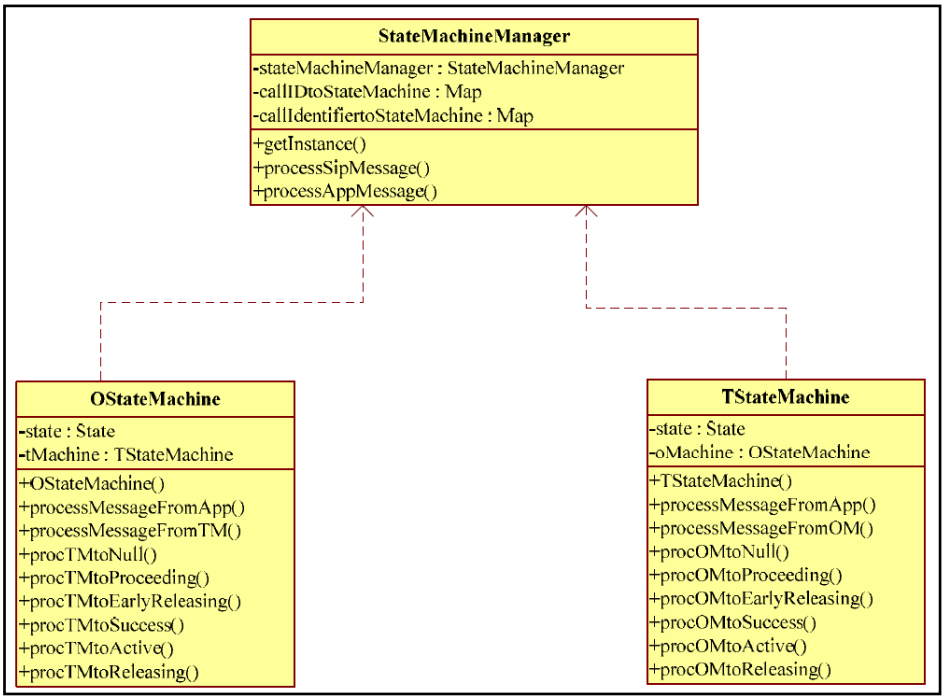


Figura A27. Máquinas de estados para el RT-WSCM

En el RT-WSCM, la clase del gestor de las máquinas de estados es la responsable de la creación y la destrucción de la máquina de estados de la llamada, y proporciona el método `getInstance` para obtener la instanciación de esta clase. El objetivo del método `processSIPMessage` es el de enviar el mensaje de petición SIP desde la pila Trillium de SIP hacia el objeto correspondiente de la máquina de estados de la llamada. El objetivo del método `processMessageFromApp` es el de enviar el mensaje de petición de la aplicación (desde la capa superior de la lógica del servicio) hacia el objeto correspondiente de la máquina de estados de la llamada.

Cada máquina de estados de las llamadas se puede dividir en dos tipos. Una es la máquina de estados de la llamada tipo O, la otra es la máquina de estados de la llamada tipo T, lo cual depende de la dirección a partir de la cual se inicia la aplicación. La máquina de estados tipo O es la responsable del procesamiento de los mensajes SIP que vienen desde la parte que llama, el módulo de la lógica del servicio, y los mensajes que provienen de la máquina de estados tipo T. La máquina de estados tipo T es la responsable del procesamiento de los mensajes SIP provenientes de la parte llamada, el módulo de la lógica del negocio, y los mensajes provenientes de la máquina de estados tipo O. Por ejemplo, cuando la aplicación inicia una llamada, se crea una máquina de estados tipo O. Y cuando la aplicación se activa desde la red, entonces se crea la máquina de estados tipo T. Tradicionalmente, una máquina de estados se diferencia por medio de un identificador de llamada único. Si el identificador de la llamada ya existía en la lista de máquinas de estados entonces se retorna la máquina de estados que existía. De lo contrario, se retorna una nueva máquina de estados creada con un identificador

único. Cuando se ha llegado al estado final, entonces la máquina de estados invoca el método correspondiente para eliminar la máquina de estados.

Trabajos Futuros

Actualmente están trabajando trabajando con motores asíncronos basados en eventos como JAIN SLEE, que es una tecnología diseñada no solo para el entorno de las Telecomunicaciones sino que también es capaz de integrar a los WS.

A.2. Antecedentes para el Manejo de Sesiones en los Servicios Web

A.2.1. Gestión de la Sesión y de las Transacciones en los Servicios Web utilizando SIP

En (Dong y Newmarch, 2005) se desarrolla un nuevo mecanismo para dar soporte a la Gestión de la Sesión en los WS utilizando el protocolo SIP, y en base a este mecanismo también se propone una solución simple para la gestión de transacciones para WS.

En las aplicaciones orientadas a la sesión existen dos problemas principales que necesitan ser resueltos. Uno de ellos es cómo gestionar en el lado del servidor múltiples sesiones; el otro de ellos es cómo mantener durante la comunicación el estado entre el cliente y el servidor. En este sentido, en (Dong y Newmarch, 2005) se analizaron dos métodos para la gestión de las sesiones en los mensajes SOAP. El primer método consiste en la integración de la gestión de la sesión en los Servidores Web; este es el caso del software Zap, del proyecto REGNET (Tsenov, 2002), que, aunque proporciona una funcionalidad sencilla para el desarrollador y el administrador del sistema, presenta una clara desventaja al hacer parte del servidor Apache, y, por tanto, no permitir la cooperación con otros servidores Web para la gestión de la sesión. La segunda solución, provista por Microsoft Corporation, consiste en la utilización del encabezado SOAP para proveer servicios con estado; sin embargo, esta versión de una sesión SOAP se basa en el mecanismo de las cookies, para lo cual el desarrollador tiene que asegurarse con cada petición de que la aplicación cliente trate las nuevas etiquetas SOAP como cookies HTTP y las envíe de regreso al servidor, con lo cual a su vez se limita el transporte de SOAP sobre HTTP, y por tanto la cooperación de SOAP con otros protocolos de transporte.

Para la demostración del mecanismo de gestión de la sesión se implementó un Sistema de WS de Venta de Videos en Línea: un usuario de WS puede establecer una sesión con el sistema (luego de que la sesión se configura, la aplicación cliente obtiene un identificador único de sesión que es almacenado por el servidor SIP, así como también es introducido en el encabezado de cada mensaje SOAP transmitido entre la aplicación cliente y el servidor) y luego invocar los WS del proveedor; luego de que el usuario finaliza todas las actividades de compra, por medio de los WS, puede terminar dicha sesión (el identificador único de la sesión se elimina tanto del servidor SIP como del servidor del WS). Este mecanismo involucra la adición de la información de la sesión en el encabezado SOAP, y la utilización de SIP para el manejo del estado de la sesión de los WS. De esta manera, se utiliza un protocolo estándar para generar el identificador de la sesión, gestionar la sesión para los WS, y para proveer extensibilidad en el tema de la seguridad.

Para la demostración del mecanismo de gestión de transacciones en los WS se implementó un Sistema de WS de Banca electrónica: en él se desarrolló la gestión de la transacción en la transferencia de dinero entre dos bancos; sin embargo, el identificador único de sesión utilizado en el mecanismo de gestión de la sesión no puede ser usado para identificar una transacción entre dos bancos debido a que el cliente puede crear muchas transacciones dentro de una sesión. Por lo tanto, para llevar a cabo la gestión de transacciones en los WS se necesita utilizar un gestor de transacciones, responsable de la coordinación de todas las actividades de los participantes en una transacción. En este mecanismo se sigue utilizando SIP para gestionar el estado de la sesión, y se introduce un gestor de transacciones, el cual es un WS que necesita comunicarse con el servidor SIP para controlar las transacciones entre los participantes.

A.2.2. Gestión Confiable de la Sesión en los Servicios Web utilizando SIP

En (Rapeeporn y Benchaphon, 2007) se emplea SIP y algoritmos de anillo para manejar sesiones confiables entre un cliente o solicitante de un servicio y un proveedor WS. SIP se utiliza en los Agentes de Usuario Servidores (User Agent Server, UAS) SIP para manejar una sesión única, por cada solicitud al servicio. La información de la sesión se mantiene en cada UAS y se utiliza entre el solicitante del servicio y los WS. Al mantener la seguridad en la sesión la información de sesión se difunde (multicast) a los diferentes UAS que estén conectados en la red, en forma de anillo virtual. Cuando el UAS principal falla, se utiliza un algoritmo de anillo para elegir el segundo UAS con la mayor prioridad.

Trabajo relacionado

La gestión de la sesión es importante para identificar a un cliente que desee consumir instancias de WS. La gestión de una sesión debería incluir los procesos de inicio, incorporación, retiro, terminación y consulta de las condiciones de la sesión a través de la red. La gestión de la sesión utilizando la técnica de las cookies esta soportada por la mayoría de los navegadores. Sin embargo, no se deberían ingresar en un archivo de una cookie los datos únicos tales como el identificador de sesión, o sesión ID. Esto se debe a que las cookies son archivos de texto plano, y son fáciles de leer utilizando un editor de texto. Microsoft también provee la gestión de la sesión para el soporte de la sesión SOAP, y por tanto para suministrar servicios con estado. Esta gestión tiene la limitación de que la versión de la sesión SOAP se basa también en el mecanismo de las cookies. Además de esto, SOAP es transportado sobre HTTP lo cual limita la colaboración entre SOAP y otros protocolos de transporte. Algunas herramientas recientes como el software Zap tienen la limitante de que sólo colaboran con el servidor Web Apache para la gestión de la sesión. Un artículo de Hada y Maruyama presenta un protocolo para la autenticación de sesiones para los WS. Su trabajo se centra en que cada participante no tiene un conocimiento previo de todas las partes que participan en la sesión.

SIP se trata de una nueva elección para adicionar la gestión de la sesión a los mensajes SOAP que se utilizan para invocar y consumir WS.

Arquitectura del Sistema

La arquitectura del sistema, Figura A28, está constituida por cuatro UAS SIP, donde uno de ellos es el servidor líder o primario. A su vez, se tienen WS y Agentes de Usuario Clientes (User Agent Client, UAC) SIP.

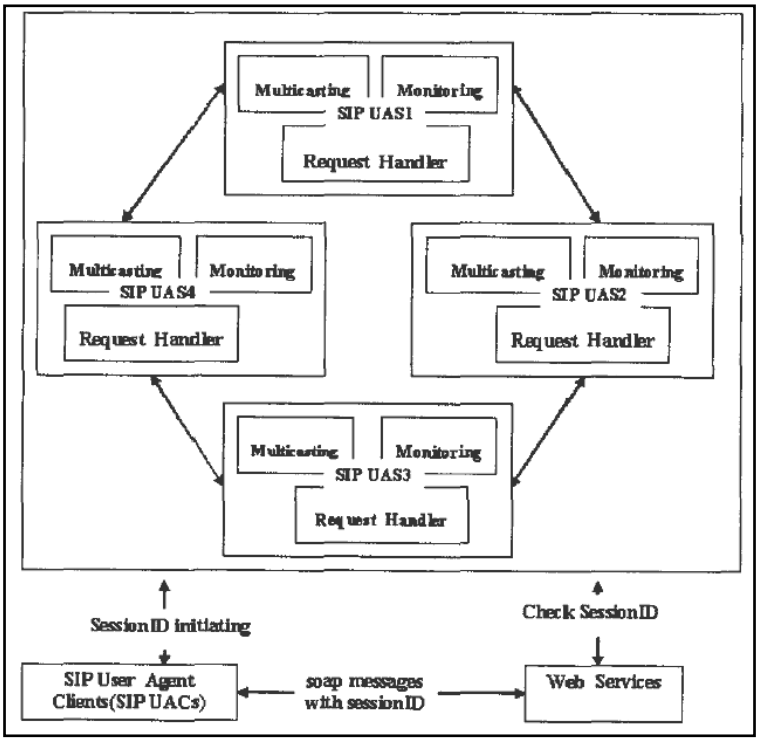


Figura A28. Arquitectura del Sistema para la Gestión Confiable de la Sesión en los Servicios Web utilizando SIP

- **UAS**

Un UAS es un servidor que genera un identificador único de sesión para cada UAC; el formato de un identificador de sesión es: `YYYYMMDDhhmmss@<UAC IP address>`, donde `<UAC IP address>` es la dirección IP del cliente. Cuando un UAC desea invocar un WS, el UAC necesita enviar una petición al UAS primario o líder. Al establecerse la conexión, el UAS líder mantiene y monitorea la sesión entre el UAC y el WS por medio de la utilización de tres módulos: el manejador de peticiones, el servicio de multidifusión y el sistema de monitoreo. El UAS se comunica con el UAC a través de mensajes SIP.

- **UAC**

Un UAC es una aplicación cliente que permite a los usuarios invocar un servicio por medio del envío de mensajes de petición hacia el UAS.

- **Manejador de Peticiones**

El modulo del Manejador de Peticiones está compuesto de tres clases: el manejador de la petición del cliente, el manejador de la petición del WS, y la tabla de sesiones.

El manejador de la petición del cliente se utiliza para manejar la petición de un mensaje SIP desde un UAC. Luego el manejador procesará y generará los mensajes de respuesta SIP o finalizará los mensajes SIP de retorno al UAC. El manejador de la petición del WS es responsable de aceptar las conexiones desde un WS, manejar y comprobar el `sessionID` de la petición del WS. La tabla de sesiones es una tabla de registro para que el UAS almacene la información actual de la sesión, como por ejemplo un `sessionID`, la dirección IP del WS y el puerto del WS. Una vez se ha establecido la sesión entre el UAC y el UAS, el UAS registrará el `sessionID` en su tabla de sesiones para usos futuros. La información que formará la `sessionID` es "@ dirección IP del cliente", que es enviada con el mensaje de petición SIP. Un UAS también registra el momento en el que el UAC establece la sesión. Se utiliza el campo "o" de SDP y las Representaciones de Fecha y Tiempo de la ISO 8601. El formato de la `sessionID` es:

```
YYYYMMDDhhmmss@<UAC IP address>
```

El resto de los datos de la tabla de sesiones son la dirección IP del WS y sus puertos. Esta información la envía el WS al UAS. El UAS luego la utilizará en el caso de la terminación de la sesión.

- **Servicios Web**

En la arquitectura del modelo propuesto se utilizan WS genéricos que dan soporte a la interacción entre clientes u otros WS, de una manera interoperable, a través de la red. La interfaz de un WS es descrita por la WSDL y el WS se comunica con quien lo solicita o con el UAC por medio de mensajes SOAP. Antes de que un WS permita a un UAC la invocación de sus servicios, tiene que comprobar el identificador de la sesión con el UAS SIP.

Implementación

- **Inicio de la Sesión**

Se asume que cada UAC y cada WS conocen la existencia del UAS en el sistema. El UAS de más alta prioridad es el líder. Cuando un UAC quiere invocar un WS, necesita preguntarle al UAS líder el `sessionID`, que es único. El proceso de inicio de sesión entre el UAS y el UAC se realiza por medio de los procesos que lleva a cabo el protocolo. De esta manera se establece la sesión entre el UAC y el UAS. Ahora el UAC ha obtenido una identificación única de sesión (el `sessionID`), la cual también es registrada por el UAS en su tabla de sesiones. Por ejemplo, asumir que la dirección IP del UAC es 202.44.40.13, el establecimiento de la sesión se da a las 5:06:30 am el 19 de diciembre de 2006. Entonces el `sessionID` será: `20061219050630@202.44.40.13`.

- **Servicio de comunicación**

En la implementación, y para demostrar el modelo propuesto en (Rapeeporn y Benchaphon, 2007), se utiliza el ejemplo de un sitio Web de comercio electrónico que incluye: el UAC, que es una tienda de compras, el WS 1 que es un proveedor de WS, y el WS 2, que es un WS de envíos. El proceso de negocio inicia cuando una orden de compra se envía al proveedor de WS.

El proveedor de WS entonces selecciona el WS de envío apropiado y le envía una petición. Cuando el WS de envío recibe la petición, éste la procesará y enviará de regreso una respuesta.

La Figura A29 muestra los pasos de comunicación del servicio. Los primeros dos pasos (1 y 2) se describieron en la sección del mecanismo de inicio. El UAS utilizará el manejador de peticiones del WS (ya descrito) para responder por medio del WS. Cuando el UAC ha obtenido el `sessionID` desde el UAS, compondrá ese `sessionID` en un encabezado SOAP, y luego lo enviará para invocar al WS 1 (paso 3).

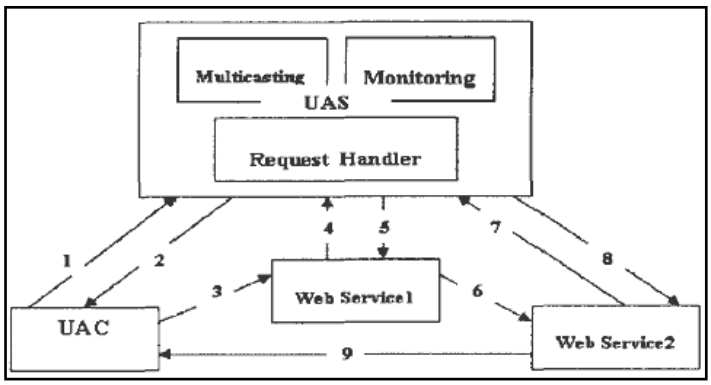


Figura A29. Comunicación en el Servicio

Al recibir el mensaje, al WS 1 se le pide que sea parte de la sesión. Entonces el WS 1 envía un mensaje, que tiene la `sessionID` del UAC, de "Check SessionID" al UAS (paso 4). Si el `sessionID` es el misma, entonces el UAS le responde con el mensaje "session has setup" (paso 5). Entonces, el WS 1 registrará el `sessionID`. Si el WS 1 necesita invocar otro servicio como el WS 2, entonces enviará un mensaje SOAP (con el `sessionID` en el encabezado SOAP) al WS 2 (paso 6). El WS 2 comprobará esta `sessionID` con el UAS y obtendrá la respuesta (paso 7 y 8). Si el WS 2 está seguro de que el `sessionID` es el mismo, entonces también lo registrará. Luego, este puede enviar un mensaje al UAC (paso 9). El WS 2 puede estar seguro que el UAC es un participante válido de la sesión.

- **Terminación de la Sesión**

Cuando un UAC culmina sus actividades con el WS, tiene que terminar la sesión. Primero el UAC envía un mensaje "BYE" (BYE@UAC IP address) al UAS líder. Luego de que el UAS lo recibe le informa a los WS, y a otros UAS, que la sesión se termino. La tabla de sesiones se actualiza.

A.2.3. WS-Addressing

Estrictamente hablando, la gestión de las sesiones no ha estado del todo ausente de SOA. Por ejemplo, con WS-Addressing (Box, y otros, 2004) se ha incluido el soporte a la sesión como un elemento opcional. Este es un estándar maduro de la W3C, utilizado para enrutar los mensajes SOAP.

Este es un estándar de servicios Web importante para el direccionamiento del servicio en una vía. Define dos estructuras interoperables que conducen información, típicamente proporcionada por protocolos de transporte y sistemas de mensajería. Estas estructuras normalizan la información subyacente y la convierten en un formato uniforme, que puede ser

procesado independientemente del transporte o de la aplicación. Las dos estructuras son: la Referencia del Punto Final (Endpoint Reference, EPR) y los Encabezados de Información de los Mensajes. Un punto final de un WS es una entidad, procesador, o recurso al cual puede hacerse referencia y hacia donde pueden dirigirse los mensajes de los WS. Las EPR pueden conducir la información necesaria para identificar/referenciar un punto final de un WS, con una resolución mejorada que va más allá de las direcciones URI o IP tradicionales, y pueden utilizarse en variadas y diferentes formas: las referencias de los puntos finales son adecuadas para la conducción de la información necesaria para acceder al punto extremo de un WS, pero también se utilizan para proporcionar direcciones a los mensajes individuales enviados hacia y desde los WS. Para tratar este último caso de uso, esta especificación define una familia de encabezados que permiten el direccionamiento uniforme de mensajes, independientemente del transporte subyacente. Estos encabezados conducen las características del mensaje extremo a extremo, incluyendo el direccionamiento para los puntos finales, tanto de la fuente como del destino, así como la identidad del mensaje.

Ambas estructuras están diseñadas para ser extensibles y reutilizables, por lo que otras especificaciones pueden construir y hacer uso de las EPR y de los encabezados de los mensajes. No obstante, WS-Addressing condiciona las sesiones a direcciones SOAP particulares y funciona solamente para las SOA que utilicen WS-Addressing. Es decir, que el modelo de sesión de WS-Addressing provee el acoplamiento entre la información del punto extremo del WS y los datos de sesión, lo cual es análogo a las referencias a objetos en sistemas de objetos distribuidos. Pero las SOA basadas en WS a menudo no lo hacen, ya que en lugar de ello pueden ser localizadas a través de URL, y por otra parte, el direccionamiento se necesita realmente sólo para los WS que a traviesan transportes que no son HTTP (Dornan, 2007).

A.2.4. WS-Coordination

El grupo WS-TX (Web Services Transaction), que se encuentra dentro de Oasis, estaba encargado de desarrollar 3 estándares: WS-Coordination (Oasis, 2009a), WS-AtomicTransaction (Oasis, 2009b) y WS-BusinessActivity (Oasis, 2009c), los cuales se aprobaron en Abril del 2007.

WS-Coordination (Dornan, 2007) es el estándar base, dirigido hacia la coordinación de metadatos. En lugar de especificar cómo se pueden coordinar los WS, este estándar les proporciona un marco para la transferencia de información acerca de la coordinación. En la parte superior de WS-Coordination, y estratificados, hay dos estándares que dirigen la coordinación en sí: WS-AtomicTransaction y WS-BusinessActivity. El primero de los estándares dirige los servicios punto a punto, y relativamente simples, y el último trata con aplicaciones más complejas.

Esta especificación describe un marco de trabajo para un servicio de coordinación (o coordinador), el cual consiste en los siguientes servicios:

- Un servicio de activación: con una operación que le permite a una aplicación crear una instancia de coordinación o contexto.
- Un servicio de registro: con una operación que le permite a una aplicación registrarse para el uso de los protocolos de coordinación.

- Un conjunto de protocolos de coordinación de tipo específico.

Las aplicaciones utilizan el servicio de activación para crear el contexto de coordinación para una actividad. Una vez que una aplicación adquiere el contexto de coordinación éste se envía, a través de cualquier medio apropiado, hacia otra aplicación. El contexto tiene la información necesaria para registrarse en la actividad, especificando el comportamiento de la coordinación que la aplicación seguirá. Adicionalmente, una aplicación que reciba un contexto de coordinación puede utilizar el servicio de registro de la aplicación original, o puede utilizar uno de los que especifique, por interposición, un coordinador de confianza. De esta manera, una colección arbitraria de WS puede coordinar su operación de acoplamiento.

Por sí mismo, este estándar no define todas las características requeridas para una solución integral. WS-Coordination es un bloque de construcción que se utiliza en conjunto con otras especificaciones y protocolos específicos de aplicación para adaptarse a una amplia variedad de protocolos relacionados con las operaciones de WS distribuidos.

Con WS-Coordination los WS tienen que manejar ellos mismos todos los contextos, por lo que, en lo concerniente a las sesiones, probablemente esto limitará a WS-Coordination a aplicaciones relativamente simples. Además, los estándares del WS-TX (Oasis, 2009d) también están limitados a trabajar con SOAP.

A.2.5. WS-Context

Namespace (Espacio de Nombres)

La URI del espacio de nombres XML que deben utilizar las implementaciones de esta especificación es: `http://docs.oasis-open.org/ws-caf/2005/10/wsctx`

Tabla A2. Espacio de Nombres de WS-Context

Prefijo	Espacio de Nombres
wsctx	http://docs.oasis-open.org/ws-caf/2005/10/wsctx
ref	http://docs.oasis-open.org/wsrm/2004/06/reference-1.1
wsdl	http://schemas.xmlsoap.org/wsdl/
xsd	http://www.w3.org/2001/XMLSchema
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
wsrm	http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd
soap	http://schemas.xmlsoap.org/wsdl/soap/
tns	http://docs.oasis-open.org/ws-caf/2005/10/wsctx

Arquitectura

Una ***actividad*** representa la ejecución de una serie de interacciones relacionadas con un conjunto de WS; estas interacciones se relacionan a través del contexto. Una actividad es una agrupación conceptual de servicios que cooperan para realizar algún trabajo; un contexto es la manera concreta en la cual tiene lugar esta agrupación. La noción de una actividad se utiliza para delimitar o definir el trabajo específico de la aplicación. La definición de lo que exactamente es una actividad, y de los servicios que ella requerirá para llevar a cabo esa labor, dependerá del entorno de ejecución y de la aplicación en la que se utiliza.

El ***contexto*** contiene información sobre el entorno de ejecución de una actividad, lo cual complementa la información en la carga útil de la aplicación. La gestión del tipo de contexto básico es facilitada por los servicios definidos en esta especificación. La especificación también proporciona interfaces de servicios para la gestión de los protocolos orientados a la sesión y para la representación de las actividades correspondientes con los contextos. La arquitectura global del contexto es jerárquica y se puede descomponer, por ejemplo es posible utilizar la estructura de contexto sin hacer referencia a ningún modelo de actividad.

El primer elemento de la especificación WS-Context es la estructura del contexto. La estructura del contexto define un modelo normal para la organización de la información de contexto. Esta estructura da soporte al anidamiento o conexión de estructuras (relaciones entre padres e hijos) para contextos relacionados, y también da soporte a mecanismos para transmitir la información de contexto por referencia o por valor. Un solo tipo de contexto no es suficiente para todas las aplicaciones; éste debe ser extensible de una manera determinada para que una especificación haga referencia a esta especificación, y los Servicios Web (Web Services, WS) deben ser capaces de ampliar el contexto, mientras lo requieran.

WS-Context define un *Context Service* para la gestión de los contextos de las actividades. El *Context Service* define el alcance (o ámbito) de una actividad, y cómo se puede hacer referencia y puede propagarse la información del contexto en un entorno distribuido. El *Context Service* utiliza el contexto para expresar la información básica acerca de la actividad. El contexto se identifica utilizando una URI. El contexto contiene la información necesaria para que múltiples WS sean asociados con la misma actividad. Esta información puede ser ampliada cuando se crea el contexto (por implementaciones de las especificaciones que hacen referencia a esta especificación), o de manera dinámica por los servicios de las aplicaciones mientras envían y reciben los contextos. Las actividades son representadas por el *Context Service*, el cual mantiene un repositorio de contextos compartidos. Siempre que los mensajes se intercambien dentro del alcance (o ámbito) de una actividad, el *Context Service* puede suministrar el contexto asociado, que entonces se puede propagar con esos mensajes.

Los contextos pueden pasarse por valor (toda la información requerida para utilizar el contexto está presente en la estructura de datos) o puede pasarse por referencia (sólo un subconjunto de la información está presente en la estructura de datos y el resto debe ser obtenida por el servicio receptor). Con el fin de dar soporte al paso por referencia, WS-Context define un *Context Manager Service* opcional al cual el receptor de un contexto por referencia puede consultar, para obtener los contenidos del contexto. Este *Context Manager Service* puede ser el mismo *Context Service*, pero dentro de WS-Context no existe ningún requerimiento para esto.

- ***Invocaciones a las Operaciones de los Servicios***

El cómo se invocan los servicios de las aplicaciones está por fuera del alcance de esta especificación: se puede utilizar el paso síncrono o asincrónico de mensajes.

Independientemente de cómo ocurren las invocaciones remotas, la información de contexto relacionada con la actividad del remitente necesita ser referenciada o propagada. Esta

especificación determina el formato del contexto, cómo es referenciado, y cómo se puede crear.

Con el fin de dar soporte tanto a las interacciones síncronas como asíncronas, los componentes son descritos en términos del comportamiento y de las interacciones que ocurren entre ellos. Todas las interacciones son descritas en términos de los mensajes correlacionados, y esto es lo que una especificación que haga referencia a esta especificación debe abstraer a un nivel superior en las parejas de petición/respuesta.

Las fallas y errores, que pueden ocurrir cuando se invoca un servicio, se transmiten de regreso a otros servicios Web en la actividad a través de mensajes SOAP, que son parte del protocolo estándar. Para lograr esto, se utiliza el mecanismo de falla del transporte subyacente, basado en SOAP. Por ejemplo, si una operación falla porque no hay presente una actividad cuando ésta se requiere, entonces la interfaz de *callback* recibirá una falla SOAP incluyendo el tipo de la falla y los elementos de información adicional específica de la implementación, soportados por la definición por defecto de SOAP. Los tipos de fallas específicos de WS-Context son descritos por cada operación. Un tipo de fallo es transmitido como un *QName XML*; el prefijo consiste en el espacio de nombres de WS-Context y la parte local es el nombre de la falla listada en la descripción de la operación.

En la medida en que las implementaciones aseguren que los formatos de los mensajes transmitidos son compatibles con los definidos en esta especificación, no se exige cual es la forma en que se implementen los puntos finales y cómo ellos exponen las diversas operaciones (por ejemplo, a través de WSDL (Christensen, y otros, 2001)). Sin embargo, en este pliego de condiciones se proporciona por defecto una normativa vinculante WSDL 1.1. Una unión a WSDL 2.0 se considerará una vez que la norma sea más general y con el apoyo disponible. Sin embargo, en esta especificación se provee por defecto un *binding* (o enlace) WSDL 1.1 que sirve de norma. Una *binding* a WSDL 2.0 se considerará una vez que el estándar esté disponible y sea soportado, de una manera más general.

Tener en cuenta que esta especificación no asume, que para las interacciones de los mensajes, se tengan que utilizar mecanismos de entrega fiable de mensajes. Así pues, las medidas que se adopten, si un mensaje no es entregado, o si no se recibe una respuesta, pueden depender de la implementación.

- ***Relación con la WSDL***

En donde quiera que se utilice a WSDL en esta especificación, se hace con mensajes en una vía (unidireccionales) y con *callbacks* (retorno de llamadas). Este es el estilo por norma. Son posibles otros estilos de enlace o *binding* (definidos quizás por especificaciones que hacen referencia a esta especificación), aunque podrían tener estilos de reconocimiento y mecanismos de entrega distintos. La definición de estos estilos está por fuera del alcance de WS-Context.

Tener en cuenta que donde se de soporte a esos componentes respectivos, las implementaciones compatibles deben adecuarse a la WSDL dada por norma y definida en la especificación. Para los componentes opcionales en la especificación se REQUIERE de la

conformidad con la WSDL solamente en los casos donde los componentes respectivos son soportados.

Para mayor claridad, la WSDL se muestra en una forma abreviada en el cuerpo principal del documento: sólo se muestran los `portTypes`; también, de acuerdo con (Christensen, y otros, 2001), se define un enlace por defecto a SOAP 1.1 sobre HTTP.

- ***Convenciones al hacer las Referencias y el Direccionamiento***

Actualmente existen múltiples mecanismos, propuestos por la comunidad de WS, para el direccionamiento de los mensajes y para hacer referencia a los WS. Esta especificación difiere de las especificaciones existentes en las reglas para el direccionamiento de los mensajes SOAP; se asume que la información de direccionamiento se ubica en los encabezados de SOAP y se respetan las reglas normativas requeridas por las especificaciones existentes.

Sin embargo, el conjunto de mensajes del Contexto requiere de un mecanismo interoperable para hacer la referencia a los WS. Por ejemplo, las estructuras de los contextos pueden hacer referencia al servicio que se utiliza para gestionar el contenido del contexto. Para dar soporte a este requerimiento, WS-Context ha adoptado un modelo de contenido, abierto para las referencias a los servicios, según lo define el *Web Services Reliable Messaging Technical Committee* (Oasis, 2004a). El esquema está definido en (Oasis, 2004b) (Oasis, 2004c) y se muestra en el código a continuación.

```
<xsd:complexType name="ServiceRefType">
   <xsd:sequence>
<xsd:any namespace="##other" processContents="lax"/>
   </xsd:sequence>
   <xsd:attribute name="reference-scheme" type="xsd:anyURI"
use="optional"/>
</xsd:complexType>
```

El `ServiceRefType` se extiende por medio de elementos de la estructura del contexto como se muestra en el siguiente código.

```
<xsd:element name="context-manager" type="ref:ServiceRefType"/>
```

Dentro del `ServiceRefType`, el `reference-scheme` es la URI del espacio de nombres para la especificación de direccionamiento a la que se está haciendo referencia. Por ejemplo, si se utiliza la especificación WS-MessageDelivery (Karmarkar, y otros, 2004) el valor sería http://www.w3.org/2004/04/ws-messagedelivery. Si se utiliza la especificación WS-Addressing (Box, y otros, 2004), entonces el valor seria http://schemas.xmlsoap.org/ws/2004/08/addressing.

El esquema de la referencia (`reference-scheme`) es opcional y solo debe utilizarse si la URI del espacio de nombres, del QName de la referencia del WS, en la cual está definida, no se puede utilizar para identificar sin ambigüedad la especificación para el direccionamiento.

Los contenidos del elemento `xsd:any` contienen una referencia del servicio según lo definido por la especificación de direccionamiento referenciada. Por ejemplo, una referencia a un

Context Manager Service puede aparecer como se muestra en el siguiente código, donde `ex` es un espacio de nombres de ejemplo.

```
<wsdl:service name="MyContextManager"
wsmd:portType="wsctx:ContextManagerPortType">
   <wsdl:port name="myCtxPort" binding="ex:ctxServiceBinding">
<soapbind:address location="http://example.com/wsdl-example1/impl"/>
   </wsdl:port>
</wsdl:service>
```

El código a continuación muestra cómo se puede utilizar un elemento derivado del `ServiceRefType`, como un contenedor para hacer una referencia a un WS.

```
<wsctx:context-manager
reference-scheme="http://www.w3.org/2004/04/ws-messagedelivery">
   <wsdl:service name="MyContextService"
            wsmd:portType="wsctx:ContextManagerPortType">
<wsdl:port name="myCtxPort" binding="ex:ctxServiceBinding">
<soapbind:address
location="http://example.com/wsdl-example1/impl"/>
</wsdl:port>
   </wsdl:service>
</wsctx:context-manager>
```

El Contexto

El contexto se utiliza para incluir en la transmisión datos específicos de un protocolo, por lo general, en encabezados SOAP (aunque no es exclusivo).

La estructura básica del contexto se muestra en el código a continuación:

```
<xsd:complexType name="ContextType">
   <xsd:sequence>
<xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="context-identifier"
type=" tns:contextIdentifierType"/>
<xsd:element name="context-service" type="ref:ServiceRefType"
minOccurs="0"/>
<xsd:element name=”context-manager” type=”ref:ServiceRefType”
minOccurs=”0”/>
<xsd:element name="parent-context" type=”tns:ContextType”
minOccurs="0">
   </xsd:sequence>
   <xsd:attribute name="expiresAt" type="xsd:dateTime"
use="optional"/>
   <xsd:attribute ref=”wsu:Id” use=”optional”/>
</xsd:complexType>
```

Las especificaciones que hacen referencia a esta especificación extienden el `wsctx:ContextType`, tanto para identificar el tipo de protocolo específico, como para extender la estructura básica del contexto para que incluya elementos y atributos específicos del protocolo.

La estructura del contexto refleja alguna parte lineal, de una relación potencialmente como de un árbol, entre los contextos del mismo tipo con la hoja de una raíz.

El contexto consta de los siguientes elementos:

- Un `wsctx:contextIdentifierType` obligatorio, llamado `wsctx:context-identifier`. Este identificador puede ser entendido como un identificador de "correlación" o un valor que se utiliza para indicar que un WS hace parte de la misma actividad. El `wsctx:contextIdentifierType` es una URI con un atributo `wsu:Id` opcional. Este elemento debe ser único.
- Un elemento `wsctx:ServiceRefType` opcional, llamado `wsctx:context-service`, que identifica la autoridad emisora responsable de generar el contexto.
- Un elemento `wsctx:ServiceRefType` opcional, llamado `wsctx:context-manager`, para obtener los datos asociados con un `context-identifier` que resuelve una referencia a un *WS Context Manager*. La presencia de este punto final (o *endpoint*) es requerida si el contexto ha sido pasado por referencia y así luego se puede utilizar para obtener el valor completo del contexto. No debería estar presente este elemento si el contexto se pasa por valor.
- Un elemento `wsctx:parent-context` opcional, que contiene alguna parte de la actual jerarquía (de padres) del contexto.
- Un atributo `wsu:Id` opcional, que puede utilizarse para dar soporte a la firma o la encriptación (cifrado) de la estructura del contexto.
- El contexto puede contener información de un número incremental de servicios. La estructura del contexto se extiende a través del elemento de extensibilidad `xsd:any`, presente en el esquema por el elemento `wsctx:ContextType`.

La propagación del contexto es posible utilizando protocolos diferentes a los que utiliza la aplicación, como se muestra en la Figura A30. La especificación de WS-Context no asume un medio específico por el cual los contextos se asocian con los mensajes de la aplicación, dejando esto a la especificación que hace la referencia.

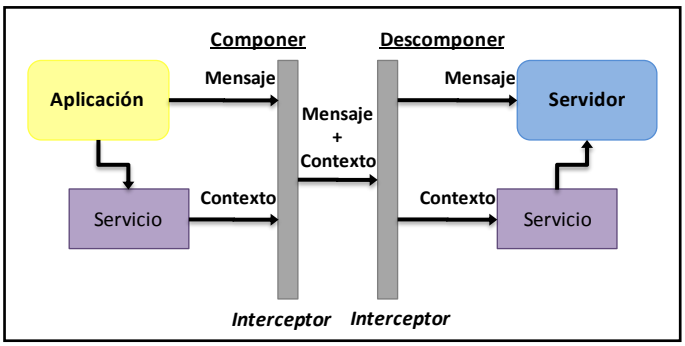


Figura A30. Servicios y flujo del contexto

Si el contexto está presente en el mensaje recibido y éste contiene un elemento *context-manager*, entonces el receptor puede utilizar dicho elemento para no hacer referencia al contexto. Por "no hacer referencia" se quiere simplemente dar a entender que se utiliza el *WS context-manager* para obtener el contexto. No se puede suponer que cualquier otra información que, en este momento, esté presente en el contexto recibido, representa el

contenido actual o total del contexto. Si el *context-manager* no está referenciado, este debería retornar todo el contenido actual del contexto, es decir, los valores correspondientes a los elementos `wsctx:ContextType` del contexto, que mantiene el *Context Service* al momento de recibir el mensaje de "no hacer referencia".

Tener en cuenta que la capacidad del gestor de contexto para devolver el contexto por valor puede ser restringida por consideraciones de seguridad, por ejemplo si quien hace la invocación no tiene los privilegios correctos.

Un contexto que se propaga por referencia, como mínimo, necesita solamente contener los elementos de `wsctx:context-identifier` y `wsctx:context-manager`. Un elemento que siempre se propague por valor, no debería contener un elemento `wsctx:context-manager`. El punto extremo (endpoint) debería retornar un fallo SOAP, con un código de error que tenga el QName correspondiente al `wsctx:InvalidContextStructure`.

Tener en cuenta que si una especificación que referencie ésta permite que un contexto, pasado por referencia, sea actualizado en el *context-manager*, entonces un servicio que mantiene una copia de un contexto pasado por referencia NO PUEDE asumir que la copia que tiene almacenada es la actual.

La elección de transmitir un contexto completo o abreviado se deja al transmisor (emisor o remitente) del contexto. Sin embargo, se espera que cuando se trata con elementos de contexto extensos, por eficiencia, se utilizará la forma de paso por referencia. Un transmisor que desee cambiar entre un modo completo y abreviado, tiene la responsabilidad de garantizar que la capacidad de "no hacer referencia" está disponible.

- ***Actividades***

Como se mencionó en la Sección de Arquitectura, una actividad se define como una colección de invocaciones a operaciones de WS, realizadas dentro de un contexto válido. Una actividad se crea, se ejecuta, y luego se termina. Un resultado o salida es el resultado de una actividad terminada. Las semánticas previstas dentro de una actividad de un WS son definidas por las especificaciones derivadas de WS-Context. Estas semánticas son indicadas por el XML QName del tipo de contexto derivado. La actividad en sí está identificada únicamente por un elemento `context-identifier`.

- ***Información de Contexto y SOAP***

Donde los mensajes requieran de contextualización (ya sea mensajes de las aplicaciones, o los mismos mensajes del protocolo WS-Context), el contexto es transportado en un bloque de encabezado de SOAP. Las especificaciones que hacen referencia a esta especificación determinan si los actores del WS-Context deben entender los contextos que llegan en los bloques de encabezados de SOAP. En el ejemplo que se muestra en el siguiente código, los destinatarios o receptores deben entender el contexto propagado con los mensajes de la aplicación. Así pues, cada bloque de encabezado SOAP que lleva un contexto, en este caso, tiene que tener el atributo "mustUnderstand" fijado en "true" ("1"), y el receptor debe entender el bloque del encabezado, codificándolo de acuerdo a su QName.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
   <soap:Header>
<example:context
   xmlns=http://docs.oasis-open.org/ws-caf/2005/10/wsctx
   expiresAt="2005-04-26T22:50:00+01:00"
   xmlns:wsdl=”http://schemas.xmlsoap.org/wsdl/”
   xmlns:soapbind=”http://schemas.xmlsoap.org/wsdl/soap/”
   xmlns:example=http://example.com/context/ soap:mustUnderstand="1">
<context-identifier>
http://docs.oasis-open.org/ws-caf/2005/10/wsctx/abcdef:012345
</context-identifier>
<context-service>
   <example:address>
http://example.org/wsctx/service
   </example:address>
</context-service>
<parent-context expiresAt="2005-04-27T22:50:00+01:00">
   <context-identifier>
http://example.org/5e4f2218b
   </context-identifier>
   <context-service>
<example:address>
   http://example.org/wsctx/service
</example:address>
   </context-service>
</parent-context>
      </example:context>
   </soap:Header>
   <soap:Body>
      <!-- Application Payload -->
   </soap:Body>
</soap:Envelope>
```

BIBLIOGRAFIA

3GPP. 2008. 3GPP TS 33.220. [En línea] 3GPP, 2008. [Citado el: 1 de Mayo de 2008.] http://www.3gpp.org/FTP/Specs/html-info/33220.htm.

BEA Systems Iberia. 2007. SPICE D2.1. Cookbook and Interface Specification of Service Enabler Components. [En línea] BEA Systems Iberia, Agosto de 2007. [Citado el: 1 de Mayo de 2008.] http://www.ist-spice.org.

Box, D., y otros. 2004. Web Services Addressing (WS-Addressing). [En línea] W3C, 10 de Agosto de 2004. [Citado en: 21 de Mayo de 2007.] http://www.w3.org/Submission/ws-addressing/.

Carvalho de Gouveia, F. y Magedanz, T. 2008. Impacts of LTE on emerging NGN Service Architectures. [En línea] Technische Universität Berlin, Fraunhofer Institut FOKUS, Mayo de 2008. [Citado el: 29 de Agosto de 2008.] http://www.comnets.uni-bremen.de/itg/itgfg521/aktuelles/fg-workshop-08052008/Gouveia_TU_Berlin_IMS.pdf.

Cheng, B., Guo, J., Meng, X., Chen, J. 2008. SIP Based Real-Time Web Services Communication Model. Computing, Communication, Control, and Management, 2008.CCCM '08.ISECS International Colloquium on.29 de Agosto de 2008, págs. 439 - 443.

Chou, W., Li, L., Feng, L. 2006. WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP. International Conference on Web Services (ICWS '06). Septiembre de 2006, págs.515-522.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. 2001. Web Services Description Language (WSDL) 1.1. [En línea] Microsoft, IBM Research, 15 de Marzo de 2001. [Citado el: 22 de Mayo de 2007.] http://www.w3.org/TR/wsdl.

Costello, R. L. 2002. REST (Representational State Transfer). [En línea] 2002. [Citado el: 15 de Enero de 2009.] http://www.xfront.com/sld001.htm.

Deason, N. 2001. White Paper SIP and SOAP, White Paper Release 1.0.[En línea] Ubiquity, Mayo de 2001. [Citado el: 7 de Marzo de 2008.] http://www.voip-info.de/res/Technologie/Ubiquity_SIP_and_SOAP.pdf.

Deason, N. y Ubiquity Software Corporation Ltd. 2000. SIP and SOAP. [En línea] IETF, Internet Draft, 30 de Junio 2000. [Citado el: 7 de Marzo de 2008.] http://www.softarmor.com/wgdb/docs/draft-deason-sip-soap-00.txt.

Dong, W., y Newmarch, J. 2005. Adding Session and Transaction Management to Web Services by using SIP. [En línea] 2005. [Citado el: 7 de Marzo de 2008.] http://jan.newmarch.name/publications/sip-soap.pdf.

Dornan, A. 2007. Tech Road Map: Oasis Takes On Web Services Session Management. [En línea] InformationWeek, 8 de Septiembre del 2007. [Citado el: 8 de Mayo de 2009.] http://www.informationweek.com/news/software/soa/showArticle.jhtml?articleID=201804545.

Fornies, H. B., Garcia, A. M., Laliena, G. C., Sanclemente, L. 2007. System for invoking Web Services by means of SIP signaling. [En línea] United States Patent Application Publication, 6 de Septiembre de 2007. [Citado el: 29 de Agosto de 2008.] http://www.freepatentsonline.com/US20070208804.pdf.

Gehlen, G., Aijaz, F., Zhu, Y., Walke, B. 2006. Mobile P2P Web Service Creation using SIP. [En línea] RWTH Aachen University, Faculty 6, Communication Networks, Kopernikusstr, 2006. [Citado el: 1 de Octubre de 2007.] http://www.comnets.rwth-aachen.de/typo3conf/ext/cn_download/pi1/passdownload.php?downloaddata=932%7C1.

González Martínez, D., Lozano Llanos, D., Rodríguez Fernández, J. A. 2008. WIMS 2.0: Ofreciendo IMS a la Web 2.0. Apertura de capacidades de comunicación mediante un servidor de Aplicaciones IMS. [En línea] 2008. [Citado el: 14 de Enero de 2009.] http://www.telecom-id.com/Actas/ProceedingsTelecomI+D-2008/pdf/025.pdf.

Hunor, D. y otros. 2007. SPICE UNIFIED ARCHITECTURE. [En línea] Nokia Siemens Networks, Junio de 2007. [Citado el: 16 de Abril de 2008.] http://www.financialtech-mag.com/_docum/iBanca_Presentacion__BEA.pdf.

IMS Laboratory, LaBRI, ENSEIRB. 2006. HomeSIP: building a smart Space with SIP Protocol. [En línea] IMS Laboratory: Laboratory of Microelectronics, Phoenix INRIA Bordeaux, School of Electrical Engineering, Computer Science and Telecommunications, Enero de 2006. [Citado el: 8 de Mayo de 2009.] http://www.enseirb.fr/cosynux/HomeSIP/.

Jähnert, J., Cuevas, A., Moreno, J. I., Villagrá, V. A., Wesner, S., Olmedo, V., Einsiedler, H. 2007. The Akogrimo way towards an extended IMS architecture.International Conference on Intelligence in Networks (ICIN2007). Emerging Web and Telecom Services: Collition or Competition. Bordeaux France 8-11 de Octubre de 2007.

Karmarkar, A, y otros. 2004. WS-MessageDelivery Version 1.0. [En línea] Oracle Corporation, Sun Microsystems Inc, Nokia Corporation, Arjuna Technologies Limited, Cyclone Commerce Inc., IONA Technologies, Enigmatec Corporation, SeeBeyond Technology Corporation, 26 de Abril de 2004. [Citado el: 22 de Mayo de 2007.] http://www.w3.org/Submission/2004/SUBM-ws-messagedelivery-20040426/.

Liu, F., Chou, W., Li, L., Li, J. 2004. WSIP – Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP. [En línea] Avaya Labs Research, 2004. [Citado el: 29 de Agosto de 2008.] http://dpnm.postech.ac.kr/mcs/papers/ALR-2004-004-paper.pdf.

López Rubio, J. y Arcas, A.O. 2007. MCU para multiconferencias en alta definición. [En línea] 1 de Febrero de 2007. [Citado el: 7 de Marzo de 2008.] https://upcommons.upc.edu/pfc/bitstream/2099.1/3775/1/55806-1.pdf.

Magedanz, T., Weik, P., Vingarzan, D., Carvalho de Gouveia, F., Wahle, S. 2008. Experiences on the Establishment and Provisioning of NGN/IMS Testbeds - The FOKUS Open IMS Playground and the Related Open Source IMS Core. [En línea] Fraunhofer FOKUS, Technical University of Berlin, Germany, 2008. [Citado el: 14 de Enero de 2009.] http://www.icin.biz/files/programmes/Session2B-2.pdf.

Oasis. 2004a. Web Services Reliable Messaging TC. WS-Reliability 1.1. [En línea] Oasis, 23 de Agosto de 2004. [Citado el: 22 de Mayo de 2007.] http://www.oasis-open.org/committees/download.php/8909/WS-Reliability-2004-08-23.pdf.

Oasis. 2004b. Schema for ServiceRefType. [En línea] Oasis, 7 de Junio de 2004. [Citado el: 22 de Mayo de 2007.] http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd.

Oasis. 2004c. Schema for WS-Reliability Protocol Headers for SOAP 1.1 & SOAP 1.2 protocols. [En línea] Oasis, 4 de Agosto de 2004. [Citado el: 22 de Mayo de 2007.] http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd.

Oasis. 2009a. Web Services Coordination (WS-Coordination). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wscoor/2006/06.

Oasis. 2009b. Web Services Atomic Transaction (WS-AtomicTransaction). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wsat/2006/06.

Oasis. 2009c. Web Services Business Activity (WS-BusinessActivity). [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 29 de Diciembre de 2008.] http://docs.oasis-open.org/ws-tx/wsba/2006/06.

Oasis. 2009d. OASIS Web Services Transaction (WS-TX) TC. [En línea] Oasis, 2 de Febrero de 2009. [Citado en: 22 de Mayo de 2007.] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx.

Parlay. 2009. Parlay. [En línea] 2009. [Citado en: Abril de 2009.] http://www.parlay.org.

Parlay X. 2009. Parlay X. [En línea] Ericsson, 2009. [Citado en: Abril de 2009.] http://www.ericsson.com/developer/sub/open/technologies/parlayx/about/parlayX.

Rapeeporn, J. y Benchaphon, L. 2007. Reliable Session Management for Web Services by Using SIP.International Joint Conference on Computer Science & Software Engineering (JCSSE2007).Mayo 2-4 de 2007, págs.161-166.

Rolan, C. y Hu, H. 2008. Defining a Service Delivery Platform Architecture using Generic IMS and SOA concepts. [En línea] Centre for Telecommunications Access and Services, School of Electrical and Information Engineering, University of the Witwatersrand, 2008. [Citado el: 18 de Abril de 2009.] http://www.icin.biz/files/programmes/Poster-1.pdf.

Tarkoma, S., Rovira, J., Postmann, E., Rajasekaran, H., Kovacs, E. 2008. Creating Converged Services for IMS Using the SPICE Service Platform.[En línea] 2008. [Citado el: 29 de Marzo de 2008.] https://www.icin.biz/files/programmes/Session7B-3.pdf.

Tekelec. 2007. Service Orchestration and Mediation: Creating a Foundation for Delivering Mixed Services. [En línea] Tekelec, 2007. [Citado el: 2 de Abril de 2008.] http://www.tekelec.com/rcenter/whitepapers/service_orchestration-mediation.pdf.

Telco 2.0. 2009. Telco 2.0. [En línea] 2009. [Citado en: Enero de 2009.] http://www.telco2.net.

Tikkanen, V. 2006. SOAP/SIP binding. [En línea] Terraventum, 2006. [Citado el: 7 de Marzo de 2008.] http://www.tml.tkk.fi/Opinnot/T-111.5360/2006/SIP.pdf.

Tsang, S., y otros. 2000. SIP Extensions for Communicating with Networked Appliances, Draft. [En línea] IETF, Noviembre de 2000. [Citado el: 8 de Mayo de 2009.] http://www.research.telcordia.com/iapp/draft-tsang-sip-appliances-do-00.txt.

Tsenov, M. 2002. Application of SOAP protocol in E-commerce Solution. 2002 First International IEEE Symposium, 2002, págs: 59 - 62.

Villagrá, V. A. y Wesner, S. 2007. AKOGRIMO Mobile Grids: Mobile Dynamic Virtual Organizations. [En línea] 2007. [Citado el: 8 de Mayo de 2009.] http://www.coregrid.net/mambo/images/stories/Events/BIGG/session%205_villagra.pdf.

ANEXO B

ANEXO B

DESARROLLO DE SERVICIOS JAIN SLEE. HERRAMIENTAS Y PROCESOS DE DISEÑO, IMPLEMENTACIÓN Y EJECUCIÓN

El presente documento tiene como objetivo proporcionar una serie de guías básicas para la instalación y configuración de las herramientas que permitieron el diseño y la implementación de MIDDIS.

Para la instanciación de la Arquitectura de la Lógica de Mediación se seleccionó a JAIN SLEE. Dentro de las herramientas y tecnologías utilizadas para el desarrollo de servicios en JAIN SLEE por una parte se destacan: Rhino SLEE SDK, librerías JAIN SLEE, SIP y SOAP RA, FSM Tool y XDoclet de Open Cloud, y por otra parte como herramientas y tecnologías complementarias se utilizaron: Eclipse IDE, editores Java y XML, Apache Ant, y PostgreSQL.

Dentro de la comunidad Java existen diversos IDE (Eclipse, Netbeans, JBuilder, JCreator, IntelliJ, ect.) para el desarrollo de aplicaciones. Sin embargo, con el fin de reducir el tiempo de creación de MIDDIS y hacer más rápida la curva de aprendizaje en la programación de aplicaciones orientadas a eventos, Eclipse IDE provee plugins que facilitan la construcción de componentes JAIN SLEE. Dentro de los dos plugins existentes está el construido por la compañía OpenCloud (Opencloud JAIN SLEE plugin) y el dirigido por Mobicents (EclipSLEE plugin). El problema de estas extensiones radica en que solo es posible construir componentes para JAIN SLEE 1.0 (JSR 22), por lo que para hacer uso de las características mejoradas y adicionales de la especificación 1.1, se requiere de versiones que por el momento no están disponibles. No obstante, Eclipse IDE fue seleccionado por ser la herramienta gratuita más usada por los desarrolladores Java.

Cada componente desarrollado para un servicio en particular necesita ser empaquetado en un ARchivo Java (JAR, Java Archive) para finalmente ser desplegado en el servidor de aplicaciones correspondiente. Aunque existan dentro de este domino herramientas como el plugin de Maven para JAIN SLEE de Open Cloud, se usó Apache Ant y la librería slee-ant-tasks.jar de Open Cloud, para realizar el empaquetamiento de componentes JAIN SLEE. La razón de no usar la herramienta Apache Maven para realizar la construcción de los .jars desplegables y de las DU para el SLEE recae en la necesidad de usar repositorios que solamente son accesibles en línea, lo cual limita el tiempo de creación, debido a su dependencia con la disponibilidad del servicio.

Una vez el código ha sido empaquetado en una unidad desplegable, es necesario probar su funcionalidad y comportamiento. Esto se hace por medio de la principal herramienta para este Ambiente de Creación de Servicios (SCE, Service Creation Enviroment), el Rhino SLEE SDK. Es en este contenedor donde las unidades son desplegadas y donde se prueba su funcionamiento, antes de desplegar el servicio en un ambiente de producción. Si se carece del Rhino SLEE SDK, no es posible comprobar la lógica y comportamiento del servicio creado.

B.1. Rhino 2.1-03

Entre las plataformas disponibles para el desarrollo de servicios JSLEE se eligió a Rhino de OpenCloud (OpenCloud, 2010a), porque, entre otras cosas, ofrece el mayor portal con información referente a esta tecnología, es considerada la mejor plataforma de JSLEE (Ruiz, 2009), y en el momento de implementar a MIDDIS era la única plataforma que cumplía completamente con la versión 1.1 de la especificación de JSLEE (1.1) (Sun Microsystems y Open Cloud, 2009), además de permitir obtener una mayor transparencia de la red subyacente a través de los RA requeridos.

Rhino da soporte a la rápida integración con sistemas externos y pilas de protocolos, y está optimizado para atender los requerimientos de desempeño y tolerancia a fallas más exigentes. La creación de servicios de Rhino se enfoca en la creación de herramientas para mitigar el riesgo en proyectos críticos. Por lo tanto, ofrece un conjunto de herramientas que facilitan un enfoque de bajo riesgo para el desarrollo y la prestación de servicios. La creación de servicios de Rhino proporciona un Entorno Federado de Creación de Servicios (Federated Service Creation Environment, FSCE), el cual hace posible el desarrollo de servicios SLEE y de Adaptadores de Recursos para la plataforma SLEE de Rhino. En el FSCE se incluyen, por una parte, herramientas para dar soporte a pruebas funcionales, de desempeño, de estabilidad, de recuperación por fallas, y por otra, también proporciona servicios de ejemplo.

En el corazón de la plataforma de Rhino está el contenedor de JAIN SLEE, el cual está diseñado y mejorado para aplicaciones dirigidas por eventos, y proporciona un modelo de programación estándar, y abierto. Rhino y JAIN SLEE son tecnologías Java, por lo tanto las aplicaciones de señalización pueden utilizar la gran familia de APIs estándares de Java. El JAIN SLEE de Rhino es un entorno de ejecución de lógica de servicios, hecho de acuerdo a los requerimientos de los estándares, el cual da soporte, gestiona, y ejecuta, de manera portable, los servicios de telecomunicaciones a nivel del portador. El modelo de programación de JAIN SLEE se ha diseñado para simplificar el trabajo del desarrollador de aplicaciones, eliminar los errores comunes de los programadores, y asegurar que se puedan desarrollar rápidamente servicios robustos.

La capacidad de integración de Rhino incluye la arquitectura del adaptador de recursos de JAIN SLEE, una arquitectura que proporciona a múltiples redes y tecnologías empresariales la capacidad de integrase, o ser “plugged into”, a la plataforma. Esta capacidad incluye:

- Adaptadores de Recursos pre-construidos, para la integración con sistemas externos comunes, por ejemplo a través de los protocolos SIP y SS7.
- Herramientas para la rápida construcción de nuevos Adaptadores de Recursos y Servicios.
- Integración con servidores de bases de datos empresariales.
- Integración con servidores de directorio.
- Integración con Servicios Web.
- Comunicaciones dúplex con servidores de aplicaciones J2EE.
- Integración de la seguridad con sistemas de directorio del Protocolo Ligero de Acceso a Directorios (Lightweight Directory Access Protocol, LDAP) y servidores web J2EE.

Los RA específicos, que se integran a la arquitectura de los RA, proporcionan la conexión entre los recursos externos y Rhino. OpenCloud ha construido diversos RA, adecuados para el despliegue en un entorno a nivel del portador, entre los cuales se encuentran los adaptadores para: SIP, el Control de Servicios IMS (IMS Service Control, ISC) (soporte a la extensión 3GPP/ISC), Diameter (interfaz de IMS), el Protocolo de Mensajes Cortos y Punto a Punto (Short Message Peer-to-Peer Protocol, SMPP) (para la mensajería de SMS), MM7 (para la mensajería de MMS), el protocolo de la Parte de Aplicación de CAMEL (CAMEL Application Part, CAP) (para la El Sistema de Señalización Número 7, SS7), el protocolo de la Parte de Aplicación de la Red Inteligente (Intelligent Network Application Part, INAP) (para el SS7), el protocolo para la Parte de Aplicación Móvil (Mobile Application Part, MAP) (para el SS7), HTTP (para integraciones empresariales), SOAP (WS), J2EE (para integraciones empresariales), LDAP (para integración con servicios de directorio), la Conectividad Java a Bases de Datos (Java Database Connectivity, JDBC) (para integración con bases de datos).

B.1.1. Preparación para la instalación de Rhino

Instalar Java (SDK)

RHINO SLEE 2.1 requiere Java SE 5 o 6. Es recomendable usar la versión más actualizada. Esta puede ser descargada desde http://java.sun.com.

Instalador: `jdk-6u20-linux-x64.bin`

La instalación solo requiere ejecutar desde la ventana de comandos el archivo .bin descargado y seguir el wizard de configuración. Los pasos mostrados a continuación resumen el proceso de instalación:

1- Ingresar como súper usuario en el Shell:

```
sudo su
```

2- Ubicarse en la dirección donde esta descargado el instalador y ejecutar el archivo .bin, antes de ello es posible que se deban configurar los permisos del archivo:

```
chmod 777 -f jdk-6u20-linux-x64.bin
./jdk-6u20-linux-x64.bin
```

3- Seguir el wizard de configuración (Aceptar la licencia, Registrar el JDK).

4- Seguir los siguientes pasos:

```
mv jdk1.6.0_20 /usr/lib/jvm
sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk1.6.0_20/bin/java" 1
sudo update-alternatives --set java
/usr/lib/jvm/jdk1.6.0_20/bin/java
```

5- Comprobar la versión de Java

```
ximena@xvelasco:~$ java -version
java version "1.6.0_20"
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) 64-Bit Server VM (build 16.3-b01, mixed mode)
```

6- Se puede fijar el javac del JDK como una "alternativa" así:

```
sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/lib/jvm/jdk1.6.0_20/bin/javac" 1
```

7- Ahora se fija la "nueva alternativa" como la real de javac en el sistema:

```
sudo update-alternatives --set javac
/usr/lib/jvm/jdk1.6.0_20/bin/javac
```

8- Para comprobar si tenemos la versión de javac 1.6.0_20, tipeamos en la terminal:

```
~$ javac -version
javac 1.6.0_20
```

9- Configurar la variable de entorno `JAVA_HOME`:

9.1- Para todos los usuarios se necesita realizar una configuración global en los archivos `/etc/profile` o `/etc/bash.bashrc`. Se ejecuta la sentencia:

```
sudo gedit /etc/bash.bashrc
```

9.2- Se agrega al archivo abierto el siguiente texto:

```
#Java Configuration
JAVA_HOME=/usr/lib/jvm/jdk1.6.0_20
export JAVA_HOME
PATH=${PATH}:${JAVA_HOME}/bin
export PATH
```

9.3- Guardar cambios.

9.4- Reiniciar el PC y ejecutar:

```
~$ echo $JAVA_HOME
/usr/lib/jvm/jdk1.6.0_20
```

Configuración de las Características de la Red (SDK)

Antes de instalar Rhino configurar las siguientes características de red:

1- Asegurarse de que el sistema tiene una IP visible en la red.

2- Asegurarse de que el sistema puede resolver `localhost` a la interfaz de loopback, utilizando el comando `ping -c 4 localhost`:

```
~$ ping -c 4 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64
time=0.041 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64
time=0.045 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64
time=0.049 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64
time=0.050 ms
--- localhost ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.041/0.046/0.050/0.006 ms
```

3- Asegurarse de que el sistema resuelve el nombre de la máquina a una IP externa, no a la dirección de loopback, utilizando el comando `ping -c 4 NombreMáquina`.

B.1.2. Desempaquetando Rhino SDK

El Rhino SLEE es entregado como un archivo .zip para Linux/Solaris o Windows. Para empezar con la instalación de Rhino:

1- Descomprimir el archivo .zip en el directorio actual de trabajo (por ejemplo `/home/ximena/rhino/`) con el comando:

```
$ unzip RhinoSDK-2.1-03.zip
```

Esto creará el directorio de la distribución, RhinoSDK, y por lo tanto el `$RHINO_HOME` será en este caso `/home/ximena/rhino/RhinoSDK`.

2- Fijar la variable de entorno `JAVA_HOME` con la localización de la instalación del JDK de Sun.

3- Leer las notas de la presente versión del Rhino SDK:

`$RHINO_HOME/doc/index.html`: contiene información de último momento.

`$RHINO_HOME/doc/CHANGELOG`: resume los cambios realizados entre las versiones previas de Rhino y la presente.

B.1.3. Ejecución del Rhino SDK

Iniciar el Rhino SDK

Para iniciar el Rhino SLEE SDK, ejecutar el script `start-rhino.sh` en el shell, el cual se encuentra en el directorio del `$RHINO_HOME`.

Podría ser útil modificar la visibilidad de la ventana de comandos (utilizando la opción de redimensionar) que ejecutará Rhino SDK, de manera que los mensajes de registro sean más fáciles de ver.

Detener el Rhino SDK

Para detener el Rhino SLEE SDK, ejecutar en otra consola el script `stop-rhino.sh`, el cual se encuentra en el directorio del `$RHINO_HOME`.

Este script tiene las siguientes opciones:

```
$ ./stop-rhino.sh --help

 Usage:
       stop-rhino.sh (--nice|--terminate|--kill)
       (Terminates the Rhino SDK.)
       Options:
       --nice          - Performs a clean shutdown of the SDK via management commands.
       --terminate     - Terminates the SDK via management commands.
       --kill          - Kills the SDK's JVM via unix system commands (unix only).

For example:

$ ./stop-rhino.sh --nice
Shutting down the Rhino SDK.
Stopping the SLEE.
Waiting for SLEE to enter STOPPED state.
SLEE is in the Stopped state on node(s) [101]
Shutdown complete.
```

Por ejemplo:

```
~/rhino/RhinoSDK$ ./stop-rhino.sh --nice
Shutting down the Rhino SDK.
Stopping the SLEE.
Waiting for SLEE to enter STOPPED state.
SLEE is in the Stopped state on node(s) [101]
Shutdown complete.
```

Configurar el Rhino SDK para que utilice Postgres

- ***Instalación de PostgreSQL***

RHINO SLEE requiere de un sistema de gestión de base de datos Postgres para almacenar el estado de la memoria de trabajo. La memoria de trabajo principal de RHINO contiene el estado de ejecución de los nodos, el estado de configuración de los adaptadores de recursos, perfiles, etc. La base de datos Postgres solo suministra un respaldo de la memoria de trabajo del servidor RHINO.

RHINO SLEE ha sido probado sobre las versiones 7.4.*, 8.1.* y 8.3.*. Este gestor de base de datos puede instalarse en una maquina independiente, aunque generalmente es instalado sobre el mismo equipo. A continuación se mencionan los pasos para la instalación.

1- Asegurarse de que los repositorios de los paquetes y los programas estén actualizados por medio de los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade --show-upgraded
```

2- Ingresar en el shell los siguientes comandos para realizar una instalación completa de PostgreSQL: las dependencias requeridas, el cliente y servidor de la base de datos, algunos scripts de utilería, la aplicación pgAdmin para administrar la base de datos, y algunos paquetes que proporcionan funcionalidades adicionales.

```
sudo apt-get install postgresql postgresql-client postgresql-
contrib pgadmin3
```

Se confirma que la instalación terminó adecuadamente:

```
ximena@xvelasco:~$ psql --version
psql (PostgreSQL) 8.4.3
incluye soporte para edición de línea de órdenes
```

Es decir que se va a utilizar la versión psql (PostgreSQL) 8.4.3 .

3- Cambiar la contraseña del usuario administrador:

Ahora necesitamos establecer la contraseña del usuario administrador postgres. Se digita la siguiente línea en el terminal:

```
sudo su postgres -c psql
ALTER USER postgres WITH PASSWORD 'postgres';
\q
```

Cambia la palabra password por postgres:

```
~$ sudo su postgres -c psql
[sudo] password for ximena:
psql (8.4.3)
Digite «help» para obtener ayuda.
postgres=# ALTER USER postgres WITH PASSWORD 'postgres';
ALTER ROLE
postgres=# \q
```

Esto altera la contraseña dentro de la base de datos, y regresa al root del shell mediante el comando `\q`. Ahora necesitamos hacer lo mismo para el usuario Linux postgres:

```
sudo passwd -d postgres
sudo su postgres -c passwd
```

Aparecerá un prompt y se introduce la misma contraseña que se puso antes.

```
~$ sudo passwd -d postgres
passwd: se ha cambiado la información de expiración de la
contraseña
~$ sudo su postgres -c passwd
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
```

4- pgAdmin (http://www.pgadmin.org/)

De ahora en adelante se puede usar pgAdmin o el terminal para administrar la base de datos como el usuario postgres. A continuación se instala el PostgreSQL Admin Pack, que proporciona funcionalidad adicional en relación a la gestión remota a través de herramientas como pgAdmin, permitiendo, por ejemplo, llevar un mejor registro y monitoreo de la base de datos.

Ejecutar lo siguiente desde la línea de comandos en el terminal:

```
sudo su postgres -c psql <
/usr/share/postgresql/8.4/contrib/adminpack.sql
```

- ***Configurar Conexiones TCP/IP***

El servidor acepta por defecto conexiones de sockets TCP/IP, ya que es una versión posterior a la 8.0. Verificar en `/etc/postgresql/8.4/main/postgresql.conf` que el servidor se encuentre configurado de forma que escuche todas las direcciones IP (es decir, se va a permitir conexiones remotas): `listen_addresses = '*'`. Dejar el puerto `5432` para el servicio.

- ***Configurar el control del acceso***

La configuración de la lista de acceso permite decirle a PostgreSQL cual método de autentificación usar, y establecer relaciones de confianza para ciertas máquinas y redes. Hay que editar el archivo `/etc/postgresql/8.2/main/pg_hba.conf`.

El servidor tiene por defecto configurado el control de acceso para las conexiones locales, necesario para el caso en donde Rhino y PostgreSQL se instalan en el mismo equipo. Verificar que en el archivo `/etc/postgresql/8.4/main/pg_hba.conf` se encuentre una configuración similar a la siguiente:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               md5
```

Si se han realizado cambios a los archivos de configuración de PostgreSQL mencionados, es necesario reiniciar completamente el servidor PostgreSQL para aplicar dichas modificaciones. Indicarle al servidor que cargue nuevamente los archivos de configuración no habilita el funcionamiento de la red TCP/IP, como ocurre cuando la base de datos es inicializada.

Para reiniciar Postgresql:

```
/etc/postgresql/8.4/main$ sudo /etc/init.d/postgresql-8.4
restart
[sudo] password for ximena:
 * Restarting PostgreSQL 8.4 database server
[ OK ]
```

- ***Configuración de Variables***

Por defecto, el Rhino SLEE SDK utiliza su propia base de datos Derby embebida. Para utilizar PostgreSQL en lugar de Derby se deben realizar los siguientes cambios:

1- Editar `config_variables`

Editar la siguiente configuración de variables en el archivo `$RHINO_HOME/config/config_variables` para que apunte a la configuración del PostgreSQL (por ejemplo: `RHINO_HOME` = `/home/ximena/rhino/RhinoSDK`).

```
# Management database settings
MANAGEMENT_DATABASE_NAME=rhino_sdk
```

```
MANAGEMENT_DATABASE_HOST=localhost
MANAGEMENT_DATABASE_PORT=5432
MANAGEMENT_DATABASE_USER=postgres
MANAGEMENT_DATABASE_PASSWORD=postgres
```

2- Editar `rhino-config.xml`

Editar el archivo `$RHINO_HOME/config/rhino-config.xml`.

Comentar la entrada de la base de datos de Derby:

- Encontrar la configuración de Derby, bajo el título `'Begin Derby-specific configuration'` y comentar esta sección:
  - En frente de la entrada `<memdb>`, inmediatamente debajo de la línea que se identifico anteriormente, adicionar el símbolo de inicio de comentario: `<!--`.
- Encontrar la línea que marca el fin de la configuración de la especificación de Derby, la cual está marcada con el texto `'End of Derby-specific database'`.
  - Insertar `-->` al final de la línea anterior (inmediatamente después del texto `</jdbc>`).

Descomentar la entrada para PostgreSQL:

- Adicionar una marca de fin de comentario (`-->`) al final de la línea que contiene el texto `'From here on is the configuration for PostgreSQL if you want to use that instead.'`.
- Buscar la línea que comienza por `'End PostgreSQL-specific section'`, y adicionar una marca de inicio de comentario (`<!--`) al inicio de la línea.

- ***Inicializar la base de datos de PostgreSQL***

Para inicializar la base de datos ejecutar desde el `$RHINO_HOME`:

```
./init-management-db.sh postgres
```

La primera vez que se ejecuta este comando:

```
ximena@xvelasco:~/rhino/RhinoSDK$ ./init-management-db.sh postgres
Creating database...
Using PostgreSQL 8.4.3
ERROR: no existe la base de datos «rhino_sdk»
org.postgresql.util.PSQLException: ERROR: no existe la base de datos «rhino_sdk»
        at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:1548)
        at org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:1316)
        at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:191)
        at org.postgresql.jdbc2.AbstractJdbc2Statement.execute(AbstractJdbc2Statement.java:452)
        at org.postgresql.jdbc2.AbstractJdbc2Statement.executeWithFlags(AbstractJdbc2Statement.java:337)
        at org.postgresql.jdbc2.AbstractJdbc2Statement.execute(AbstractJdbc2Statement.java:329)
        at com.opencloud.rhino.tools.SQLUtils.dropDatabase(SQLUtils.java:24)
        at com.opencloud.rhino.tools.InitManagementDB.initDatabase(InitManagementDB.java:182)
        at com.opencloud.rhino.tools.InitManagementDB.initPostgresDatabase(InitManagementDB.java:116)
        at com.opencloud.rhino.tools.InitManagementDB.main(InitManagementDB.java:87)
CREATE DATABASE
Using PostgreSQL 8.4.3
Database has been initialised.
ximena@xvelasco:~/rhino/RhinoSDK$
```

A partir de ahora, siempre se tendrá que pasar la palabra postgres para este script.

Al inicializar la base de datos se crean las tablas que utiliza Rhino SLEE y la tabla usada por el ejemplo de conectividad del protocolo SIP (RA y servicios) proveído por OpenCloud. Los

comandos SQL ejecutados en esta acción, teniendo en cuenta que la base de datos utilizada es PostgreSQL, se muestran a continuación:

```
create table versioning (
  name text not null primary key,
  application text not null,
  ocbb text not null,
  description text not null,
  version integer not null
);
GO
CREATE TABLE keyspaces (
  dbid text not null,
  key_id text not null,
  mode int4 not null,
  timeout int4 not null,
  table_name text not null,
  primary key (dbid, key_id, mode)
);
GO
CREATE TABLE timestamps (
  dbid text not null primary key,
  era int8 not null,
  last_update int8 not null
);
GO
create table registrations ( -- for SIP
location service
  sipaddress varchar(80) not null,
  contactaddress varchar(80) not null,
  expiry bigint,
  qvalue integer,
  cseq integer,
  callid varchar(80),
  flowid varchar(80),
  primary key (sipaddress, contactaddress)
);
GO
```

Iniciar Rhino SDK con la nueva configuración de base de datos PostgreSQL. Nuevamente en consola ejecutar `start-rhino.sh` dentro de la carpeta del `$RHINO_HOME`.

B.1.4. Desinstalación del Rhino SDK

Detener el SLEE y borrar el directorio en el cual se instaló el Rhino.

B.1.5. Herramientas para la Gestión de Rhino

Consola de Línea de Comandos (*rhino-console*)

Este es un shell de línea de comandos el cual da soporte tanto a comandos interactivos como a los archivos por lotes para gestionar y configurar el Rhino SLEE. Puede ser ejecutada por medio de los scripts "`client\bin\rhino-console.bat`" o "`client/bin/rhino-console`", y tiene disponible ayuda en tiempo de ejecución al ingresar la palabra `help`.

Estadísticas y Monitoreo

Este es un cliente basado en GUI para el monitoreo de la información estática proporcionada por Rhino. Esta interfaz puede ser ejecutada por medio de los scripts "`client\bin\rhino-stats-gui.bat`" o "`client/bin/rhino-stats -g`".

Consola Web (*web-console*)

Por defecto, el Rhino SDK lanza un cliente de administración basado en Web, Figura B1, el cual está disponible en `http://localhost:8066`. La consola Web proporciona una interfaz de usuario HTML para la gestión de Rhino. Esta utiliza JAAS para proporcionar seguridad y permitir acceso a las funciones a usuarios autorizados. Entre dichas funciones se encuentran: despliegue de aplicaciones, provisión de los perfiles SLEE, visualización y uso de parámetros, y configuración de los adaptadores de recursos, entre otras.

```
Username: admin
Password: password
```

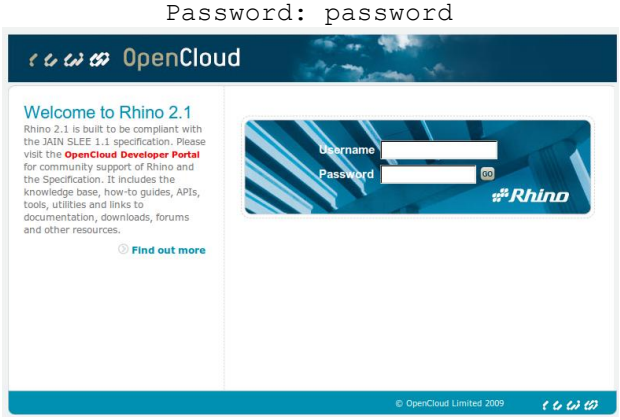


Figura B1. Consola Web de Rhino

B.2. FSM TOOL

La FSM Tool (OpenCloud, 2009) es una herramienta liviana diseñada para simplificar la creación de servicios en el Rhino SLEE. El desarrollador ya no tiene que codificar a mano y mantener la compleja jerarquía de máquinas de estados Java. En lugar de ello, las máquinas de estados de los servicios son definidas formalmente, en un lenguaje de dominio específico, a fin de que el comportamiento de la máquina de estados pueda implementarse automáticamente, y esté lista para su uso en Rhino. Desde la perspectiva del desarrollador, la herramienta se asegura de que los artefactos de la especificación y la implementación permanezcan completamente precisos y sincronizados durante todo el desarrollo y el mantenimiento del ciclo de vida de cada componente.

B.2.1. Introducción

Máquina de Estados Finitos (Finite State Machine, FSM)

Una máquina de estados finitos es un modelo de comportamiento que consiste de:

- Un conjunto de estados

- Un conjunto de transiciones entre los estados
- Acciones en respuesta a eventos o transiciones

Las FSM están clasificadas como máquinas Moore o Mealy. Las Máquinas de Moore solamente utilizan acciones de ingreso (*entry actions*), y sus salidas dependen solamente de los estados. Las máquinas de Mealy utilizan solamente acciones de entrada (*input actions*), y sus salidas dependen de las entradas y los estados. Mientras que las máquinas de Moore son por lo general más simples, las máquinas de Mealy tienen pocos estados.

Máquina de Estados Finitos y Virtuales (Virtual Finite State Machine, VFSM)

Una máquina de estados finitos y virtuales es una FSM que opera en un entorno virtual, y se puede definir como "un método de especificación de software para describir el comportamiento de un sistema de control".

Una VFSM también es un modelo híbrido de máquina de estados, en el sentido de que incorpora características de las máquinas de Moore y Mealy. Las VFSMs utilizan tres tipos de acciones:

- *input* (como una máquina Mealy)
- *entry* (como una máquina Moore)
- *exit* (una característica adicional, útil para requerimientos de aplicaciones tales como la cancelación de temporizadores)

La VFSM ha evolucionado a través del desarrollo de grandes sistemas de comunicaciones. Se ha convertido en un modelo flexible y fácil de utilizar, y con un lenguaje bien definido, para el desarrollo de software. En particular, la VFSM llena el vacío entre el diseño/requerimientos/especificación y la implementación, como sigue:

1. Los desarrolladores escriben una especificación VFSM que describa de manera completa el comportamiento de un componente de software.
2. Los desarrolladores extienden directamente el modelo de comportamiento especificado, adicionando las implementaciones de las acciones definidas en la especificación.
3. Luego la VFSM crea automáticamente un ejecutable del software.

Las partes virtuales de la VFSM son sus entradas y salidas. Estas definen la interfaz entre el modelo de comportamiento y la implementación. Todas las entradas son variables Booleanas. Todas las salidas son nombres o referencias no parametrizadas. Así virtualizada, la VFSM proporciona una separación limpia entre las especificaciones del comportamiento y su implementación. Esto significa que el software y el diseño evolucionan juntos, siempre en sincronización.

- ***Especificación de la VFSM***

  Una **especificación VFSM** consta de los siguientes elementos:

  - **Estados**: un número finito de estados de la aplicación, cada uno de los cuales representa un punto al cual ha llegado la aplicación en su ejecución.

- **Transiciones**: expresiones condicionales de las entradas, que producen una transición de un estado a otro, sólo si la condición es verdadera.

- **Acciones de entrada (input)**: acciones ejecutadas en función del estado y las entradas actuales. Hacen referencia a una lógica en la implementación que es ejecutada cuando la VFSM está en un estado en particular, recibe una entrada en particular, y una condición se evalúa como verdadera.

- **Acciones de salida**: son acciones que se ejecutan cuando se sale de un estado. Son una lista de nombres que hacen referencia a una lógica en la implementación, la cual es ejecutada en el estado de origen y al inicio de una transición.

- **Acciones de ingreso (entry)**: son acciones ejecutadas en la transición hacia un estado. Son una lista de nombres que hacen referencia a una lógica en la implementación, que es ejecutada cuando se alcanza el estado de destino, al final de la transición.

- **Diccionario de salida**: nombres de las acciones de salida que se van a ejecutar. Son un conjunto declarado de acciones utilizadas en acciones de entrada, salida, e ingreso a los estados.

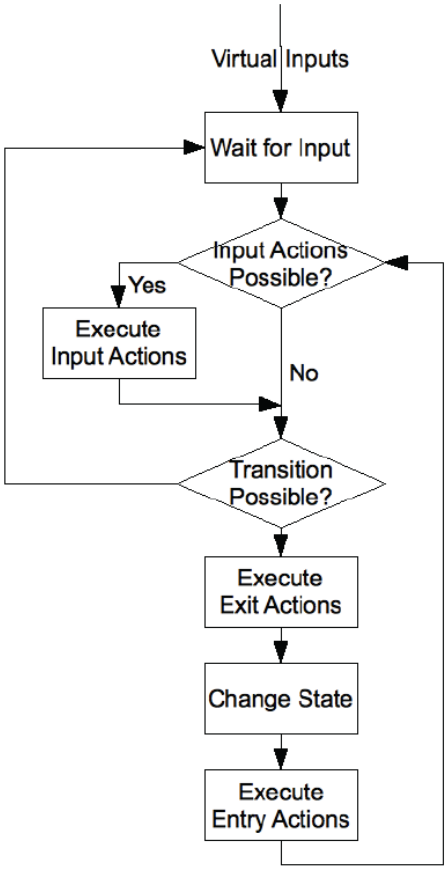
- **Diccionario de entrada**: son banderas virtuales generadas para indicar la llegada de un evento u otra condición. Es un conjunto declarado de entradas virtuales Booleanas, utilizadas en las acciones de entrada y las expresiones condicionales de las transiciones.

- **Condiciones**: lógica positiva, algebra booleana, utilizando solamente los operadores AND y OR sobre las variables booleanas, principalmente para crear un entorno virtualizado que pueda representar las sentencias de control como tablas.

- ***Funcionamiento de la VFSM***

El modelo de ejecución que se utiliza en la FSM Tool varía del modelo de ejecución estándar de la VFSM. Las acciones de la entrada (*input actions*) son evaluadas luego de cada transición no solamente al momento de la entrada que inicia la ejecución. Por una parte, esto asegura que la FSM se comporte de la misma manera sin importar cómo se fijó una entrada, y, por otra, resuelve una inconsistencia presente en la actual especificación de la VFSM.

A continuación se describe el funcionamiento de la VFSM:



1. Al inicio, la VFSM se encuentra en un estado de espera, atenta a las **entradas (inputs)**.
2. Cuando se recibe una entrada, la VFSM evalúa las **acciones de entrada** posibles **(input actions)** para el estado actual (estado de espera).
3. Si encuentra las acciones para la entrada, las ejecuta.
4. Cuando termina la ejecución del conjunto de acciones, la VFSM evalúa las posibles **transiciones**.
5. La VFSM realiza la transición de la primera condición que evalúa como verdadera.
6. Luego ejecuta las **acciones de salida** del estado de origen.
7. A continuación cambia de **estado**.
8. La VFSM ejecuta **acciones de ingreso (entry actions)** para el estado de destino.
9. Luego la VFSM comprueba si se puede ejecutar alguna de las **acciones de entrada (input actions)**.
10. Luego de esta secuencia, la VFSM realiza una comprobación para saber si hay alguna transición de estado, de alguna entrada virtual, que la VFSM haya o no fijado durante las acciones precedentes.
11. La VFSM continúa realizando las transiciones de estado, hasta que no haya más transiciones posibles, y vuelve al estado de espera.

B.2.2. Desarrollo de servicios SLEE con la FSM Tool

La FSM Tool y Rhino

Rhino es un servidor de aplicaciones, orientado a eventos, para servicios de JAIN SLEE. Un servicio típico debe manejar diferentes eventos de una o más fuentes (tales como un MSC, S_CSCF, o SMSC). Con Rhino, un manejador de eventos procesa cada evento que recibe un servicio y ejecuta acciones, es decir la lógica asociada con el evento y el estado actual de la aplicación.

JAIN SLEE no define el concepto de un estado o acción de una aplicación. El desarrollador de servicios puede escoger la manera en que mejor se implemente la máquina de estados de una aplicación. Por ejemplo, muchos desarrolladores de JAIN SLEE utilizan las APIs de JAIN SLEE directamente y codifican a mano la lógica de la máquina de estados asociada con cada estado de la aplicación y las transiciones. No obstante, la codificación a mano del modelo de comportamiento de una aplicación Rhino rompe el vínculo entre la especificación y su implementación.

La FSM Tool de OpenCloud facilita la codificación de FSM para Rhino. Esta herramienta formaliza la definición del comportamiento de la máquina de estados de una aplicación y al mismo tiempo proporciona un marco para la implementación y el mantenimiento de acciones

en Java (sin tener que codificar a mano la lógica del comportamiento de la máquina de estados). La herramienta proporciona la lógica que ejecuta acciones cuando un servicio recibe un evento. De esta forma, mientras progresa el proyecto, se mantiene el vínculo entre el diseño y la implementación de la aplicación.

Tener un enfoque formal de la especificación de la máquina de estados tiene algunas ventajas adicionales. La FSM Tool puede generar automáticamente:

- ***Documentación***: la especificación formal de la máquina de estados documenta automáticamente el modelo de comportamientos del componente de la aplicación. La FSM Tool genera automáticamente tanto una descripción tabulada como un diagrama de la máquina de estados, y las inserta en el Javadoc del SBB que se está desarrollando. De esta manera, los desarrolladores y las futuras personas encargadas del mantenimiento de las aplicaciones tendrán automáticamente componentes bien documentados.
- ***Pruebas***: las futuras versiones de la FSM Tool introducirán la generación automática de un conjunto de pruebas de unidades JUnit para el comportamiento de las máquinas de estados, y un modelo Promela de la máquina de estados con análisis estático utilizando el software de comprobación del modelo Spin.

Simplificación de la implementación de VFSM

Como ya se describió previamente, la FSM Tool utiliza una especificación híbrida de la máquina de estados denominada VFSM. Los desarrolladores especifican las VFSM para una aplicación por medio de un lenguaje fácil de usar y de dominio específico, y utilizan explícitamente la terminología ("estados", "eventos", "eventos de entrada", y "acciones de salida") del diseño de máquinas de estado.

Cada especificación VFSM de la FSM Tool define una sola máquina de estados. Esto la hace simple de entender y mantener. Las jerarquías de máquinas de estados necesarias para implementar aplicaciones complejas se componen de múltiples intercomunicaciones de las especificaciones VFSM, las cuales se comunican utilizando lógica de mapeo de entrada y salida.

Flujo de Trabajo con la FSM Tool

El siguiente diagrama, Figura B2, ilustra el flujo de trabajo de un desarrollador cuando utiliza la FSM Tool, destacando las características actuales y previstas de la herramienta.

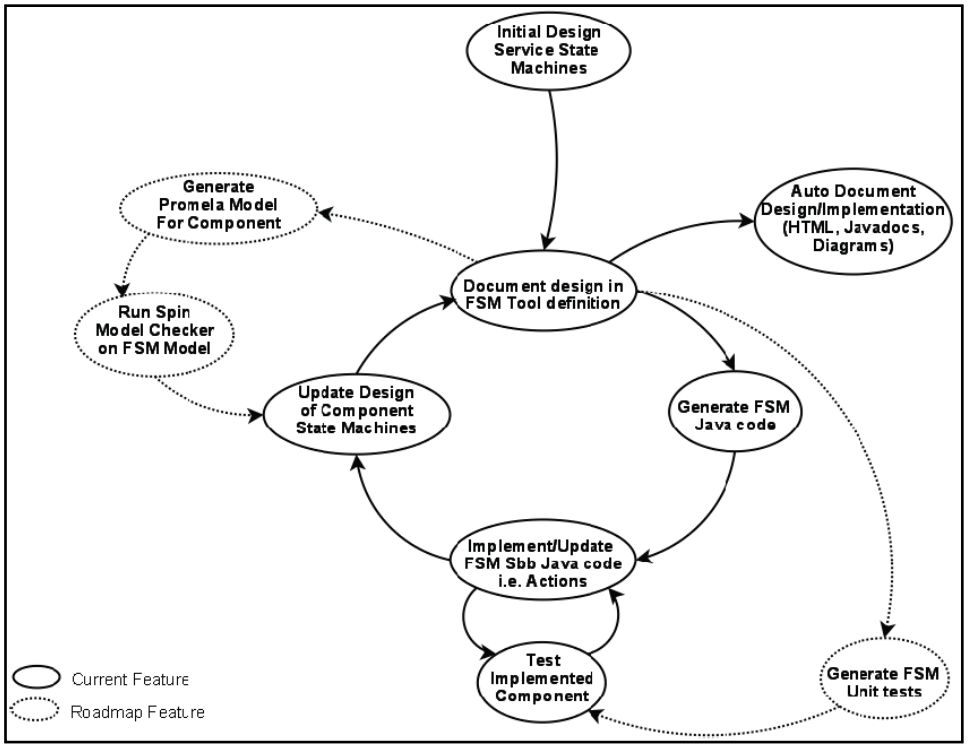


Figura B2. Flujo de Trabajo con la FSM Tool

Ejemplo: definición de las máquinas de estados de los SBB

En los servicios de JAIN SLEE, por lo general, un Bloque de Construcción de Servicios (Service Building Block, SBB) implementa un componente de una aplicación con una máquina de estados bien definida. La FSM Tool hace simple el proceso de definir la máquina de estados de un solo SBB.

Una aplicación JAIN SLEE típica incluye un árbol de entidades SBB. Estos se adaptan muy bien a un diseño de servicios basado en una jerarquía de máquinas de estados. Los SBB implementan cada máquina de estados en la jerarquía, y las interfaces del objeto local entre los SBB padre e hijo proporcionan el mecanismo para la comunicación entre los estados de las máquinas. A un objeto de un SBB solamente se le permite invocar a los objetos locales de SBB que representen entidades SBB en el mismo árbol de entidades. No está definido el comportamiento de la ejecución del SLEE si un objeto SBB intenta invocar un objeto local de un SBB que represente a una entidad SBB árbol de entidades distinto.

Instalación de la FSM Tool

A continuación se describen los pasos necesarios para empezar a utilizar la FSM Tool.

- ***Comprobar los requisitos***

Para utilizar la FSM tool se necesita:

- ✓ Rhino 2.0 o Rhino 2.1
- ✓ Apache Ant 1.7

- ✓ OpenCloud XDoclet tool: la version utilizada fue la 1.2.3-oc1, descargar desde: https://developer.opencloud.com/devportal/display/DWNLD/OpenCloud-XDoclet-DownloadPage. Los módulos de XDoclet se utilizan para la generación de los descriptores de despliegue, en base a etiquetas especiales en los archivos fuente de Java, para los componentes que se despliegan en el Rhino de OpenCloud.
  El paquete contiene ejemplos acerca de cómo utilizar los módulos de XDoclet de OpenCloud en la generación de los descriptores de despliegue definidos por las especificaciones JAIN SLEE 1.0 y 1.1, y los descriptores de despliegue para las extensiones de Rhino de OpenCloud.
  Descomprimir el archivo `opencloud-xdoclet-2.1.1.zip` en el `$RHINO-HOME`.
- ✓ GraphViz: utilizado para generar automáticamente los diagramas VFSM. Descargar para Ubuntu, desde: http://www.graphviz.org/Download_linux_ubuntu.php, los archivos: `graphviz_2.26.3-1_amd64.deb`, `libgraphviz4_2.26.3-1_amd64.deb`, `libgraphviz-dev_2.26.3-1_amd64.deb`. Es posible que el sistema requiera que se instale algún software adicional antes de poder instalar los anteriores archivos. Realizar la instalación de los archivos en el siguiente orden:

```
dpkg -i libgraphviz4_2.26.3-1_amd64.deb
dpkg -i libgraphviz-dev_2.26.3-1_amd64.deb
dpkg -i graphviz_2.26.3-1_amd64.deb
```

- ***Instalar la FSM Tool***

Para instalar la FSM Tool:

1. Descomprimir el archivo `fsmtool-0.8.5.1.zip` en el `$RHINO_HOME`. Por ejemplo:

```
~/rhino/RhinoSDK$ unzip fsmtool-0.8.5.1.zip
~/rhino/RhinoSDK$ cd fsmtool/
~/rhino/RhinoSDK/fsmtool$
```

2. Fijar las variables `rhino.home`, `slee.version`, `graphviz.dot`, y `xdoclet.lib` en el archivo `build.properties` del proyecto, en este caso **`middis`**, localizado en: `$RHINO_HOME/fsmtool/middis/build.properties`.

```
# Location of Rhino client install
client.home=/home/ximena/rhino/RhinoSDK/client

# Rhino Home
rhino.home=/home/ximena/rhino/RhinoSDK

# Graphviz
graphviz.dot=/usr/bin/dot

# XDoclet
xdoclet.lib=${rhino.home}/opencloud-xdoclet-2.1.1/libs
# XDoclet version
xdoclet.version=1.2.3-oc1
# OpenCloud XDoclet modules version
opencloud.xdoclet.version=2.1.1
```

Procedimientos de la FSM Tool

- ***Tareas de desarrollo***

Para desarrollar un componente con la FSM Tool:

- Escribir la especificación FSM: utilizar el Lenguaje de Dominio Específico (Domain Specific Language, DSL) de la VFSM para escribir un archivo de texto en el que se definen los estados, acciones, transiciones, entradas y salidas.
- Generar un SBB: utilizar la especificación que se escribió para generar el código del SBB.
- Extender el SBB generado: desplegar las clases en tiempo de ejecución, extender la clase, mapear los eventos a las entradas, terminar la fijación de las entradas duraderas, e implementar los métodos de acción.
- Compilar, empaquetar y desplegar: utilizar los métodos estándar para desplegar el SBB completo.
- Documentar y realizar el trazado: utilizar las características de la FSM Tool para documentar automáticamente el SBB y permitir el trazado en tiempo de ejecución.

- ***Especificar la máquina de estados***

Primero escribir la especificación para las máquinas de estados constitutivas. En una aplicación simple a lo sumo habrá una sola máquina de estados, pero en aplicaciones más complejas puede haber muchas máquinas de estados en una jerarquía.

La FSM Tool utiliza el DSL de la VFSM para especificar las máquinas de estados. La sintaxis de este lenguaje incluye:

- Definiciones y descripciones de los estados.
- Entradas, salidas, y acciones de ingreso asociadas con cada estado, con las condiciones que se deben cumplir antes de la ejecución.
- Transiciones, con las condiciones que se deben cumplir antes de realizar la transición.
- Un diccionario de entrada (`inputdictionary`), en el que se declaran todas las entradas (eventos de entrada) a las que hacen referencia las definiciones de los estados
- Un diccionario de salida (`outputdictionary`), en el que se declaran todas las salidas a las que hacen referencia las definiciones de los estados.

- ***Generar la clase Java del FSM SBB***

La FSM Tool utiliza la especificación FSM para generar una clase Java FSM del SBB. Para generar correctamente el código se necesita:

- Fijar las propiedades para la plantilla de generación de código FSM del SBB, de la FSM Tool.
- Configurar el archivo `build.xml` de Ant, para utilizar la *FSM Tool Ant task*:
  - Agregar al archivo `build.xml` una tarea `<vfsmgen>`, y fijar las propiedades de la tarea para el componente que se está desarrollando.
  - Ejecutar Ant en el objetivo o target que incluye la tarea `<vfsmgen>`.

- Revisar los errores que reporta la FSM Tool cuando se hace el parseo de la especificación, corregirlos, y luego volver a correr el target de Ant.

- ***Extender el SBB generado***

La especificación VFSM define el comportamiento abstracto del componente. Para hacer que funcione, se debe extender la clase Java FSM, del SBB, generada con una clase SBB que:

- Mapee los eventos recibidos a las acciones de entrada, y ejecute la máquina de estados.
- Implemente los métodos de acción que la clase generada declara abstractamente.

Seguir estos pasos para extender el SBB:

1. Desplegar las clases en tiempo de ejecución. La FSM Tool tiene un entorno de ejecución pequeño. La clase FSM generada del SBB, implementa la interfaz SbbStateMachine. Se deben desplegar las clases en tiempo de ejecución junto con el SBB, ya sea como una librería .jar o con clases que estén incluidas en el .jar del SBB.
2. Extender la clase generada.
3. Mapear eventos a las entradas de la máquina de estados. Los desarrolladores de JAIN SLEE están acostumbrados a escribir la lógica en los métodos manejadores de eventos del SBB. La FSM Tool cambia este estilo, e implementa todo el código de la lógica del servicio en los métodos de las acciones. Con la FSM Tool sólo se utilizan los métodos manejadores de eventos para el mapeo de los eventos a las entradas de la máquina de estados, llamando al método `receivedInput(Input, Object, ActivityContextInterface)`, y es posible realizar algún proceso de registro.

   Cuando se utiliza la FSM Tool, los manejadores de eventos del SBB sólo se utilizan para mapear los eventos a las entradas de la máquina de estados, y para llamar a la máquina de estados.

   Cuando se desarrolle un SBB, crear los métodos manejadores de eventos de forma similar, con el primer atributo fijado a la entrada virtual asociada.

   La FSM generada del SBB declara entradas en una clase `Enum` llamada "`Input`". Se puede extender la FSM del SBB para que directamente haga referencia a estas entradas. Por ejemplo, si la entrada declarada es `Input.hello`, cuando el servicio llama el método `receivedInput(Input, Object, ActivityContextInterface)`, el código generado por la FSM Tool fija la entrada `hello`, corre la máquina de estados, y ejecuta alguna acción.

   Si se necesita fijar más de una entrada, antes de ejecutar la máquina de estados, utilizar el método `receivedInput(Input[], Object, ActivityContextInterface)`, como sigue:

```
public void onMessageEvent(MessageEvent event, ActivityContextInterface aci) {

        Input[] inputs = {Input.hello, Input.niceWeather};
        receivedInput(inputs, event, aci);
    }
```

El método `receivedInput()` no puede ser llamado en un método de acción, esto ocasionaría una excepción en tiempo de ejecución.

4. Terminar la fijación de las entradas duraderas. Por defecto, en el diccionario de entrada de la especificación de la máquina de estados, las entradas son de corta duración o transitorias. (Las FSMs limpian automáticamente las entradas transitorias después de evaluar las condiciones de la transición). Si se tiene que utilizar entradas duraderas con un método de una acción, de manera manual se necesitará terminar su fijación, como sigue:

```
public int helloWorldAction(Object event, ActivityContextInterface aci) throws Exception {

    unsetInput(Input.niceWeather);

}
```

5. Implementar los métodos de acción. Cada acción definida en la especificación de la máquina de estados es un método abstracto en la clase FSM del SBB. Para que una acción funcione en el servicio terminado, tiene que implementarse en el SBB extendido. Los métodos de las acciones de la máquina de estados ejecutan toda la lógica asociada a la llegada de un evento.

- ***IDEs y XDoclet***

En un entorno de desarrollo integrado (IDE) como Eclipse, el diálogo de "`Override/Implement Methods`" se puede utilizar para crear los stubs o bosquejos de los métodos. Asegurarse de que solamente estén marcados los métodos de las acciones (aquellos con el sufijo `Action`).

Se recomienda la utilización de la herramienta XDoclet de OpenCloud para generar los descriptores de despliegue para los SBBs (ya que la clase generada del SBB declara los campos CMP utilizando las etiquetas de XDoclet). Si no se utiliza XDoclet se necesitará mantener a mano los descriptores de despliegue del SBB extendido. También se tendría que incluir los campos CMP `StoredState` y `EncodedVirtualInputRegister` en el descriptor de despliegue del SBB.

- ***Compilar, empaquetar y desplegar el SBB***

Una vez se haya terminado los pasos para implementar un SBB, utilizar las tareas Ant para compilar, empaquetar, y desplegar el servicio.

- ***Generar la documentación***

La FSM tool viene con comandos para generar la documentación y el trazado automáticamente.

Para generar una ilustración gráfica de la máquina de estados, configurar una tarea llamada `<vfsmgen>`, para generar el formato de archivo .dot de Graphviz.

La FSM Tool utiliza plantillas de generación de código para especificar los detalles de la implementación que genera una especificación de una máquina de estados. La salida de la generación de código no depende directamente de la FSM Tool, ya que solamente al cambiar las plantillas se pueden producir formatos de salida completamente diferentes. Esto significa que a partir de la especificación se puede utilizar la FSM Tool para generar muchos y diversos tipos de archivos de salida. Por ejemplo, la FSM Tool viene con las siguientes plantillas para la generación de javadocs y gráficas, a partir de la especificación de la máquina de estados:

- `vfsm.stg`: para generar una clase Java FSM SBB.
- `vfsm-pojo.stg`: para generar una clase Java FSM POJO para utilizar en los RAs de JAIN SLEE 1.1.
- `dot.stg`: para generar un archivo dot GraphViz.

La FSM Tool puede utilizar las plantillas proporcionadas, utilizar plantillas mejoradas para cumplir con requerimientos específicos, o puede utilizar plantillas creadas desde cero. Las plantillas están escritas en el formato del grupo de plantillas StringTemplate. Cualquier grupo de plantillas para la generación de código utilizadas por la FSM Tool debe implementar el conjunto completo de nombres de plantilla utilizados en el archivo vfsm.stg. Las implementaciones de las plantillas pueden generar cualquier salida de texto o cadenas vacías si así lo requiere el formato de salida final (por ejemplo, .java, .dot, .html, etc.).

- ***Habilitar el trazado en tiempo de ejecución***

La FSM generada del SBB se puede configurar para producir mensajes de trazado en los estados y durante las transiciones, fijando el nivel de trazado en "`Fine`" para el SBB desplegado. Una vez se fija el trazado, se escriben en el subsistema de logging de rhino mensajes de trazado como los siguientes:

```
[SbbID[name=FsmtoolExampleSbb,vendor=OpenCloud,version=1.0]:FsmToolExampleSbb]
        pkey=101:123877153487:0 VFSM[transition=startState->stopState, VIR[hello]]
```

El trazado de las acciones (así como el trazado de las transiciones) se puede habilitar mediante la fijación en "`Finest`" del nivel de trazado del SBB. Un ejemplo de salida para una acción en el registro de Rhino se muestra a continuación:

```
[SbbID[name=FsmtoolExampleSbb,vendor=OpenCloud,version=1.0]:FsmToolExampleSbb]
        pkey=101:123877153487:0 VFSM[transition=startState->stopState, VIR[hello]]
[SbbID[name=FsmtoolExampleSbb,vendor=OpenCloud,version=1.0]:FsmToolExampleSbb]
        pkey=101:123877153487:0 VFSM[action=helloWorld, state=stopState, result=executed
 successfully]
```

Configuración de la Generación de Código

- ***Fijación de los parámetros de las tareas de Ant para las plantillas de generación de código de la FSM Tool***

La FSM Tool incluye una tarea Ant para la integración de la generación del código con la infraestructura de construcción existente.

Declaración de una tarea Ant

La tarea Ant de la FSM Tool debe estar declarada en el archivo Ant `build.xml`. La FSM Tool viene con el archivo `state-machine-tool.jar`, el cual contiene la tarea Ant. Este se debe instalar en una ruta referenciada por el `taskdef` de Ant.

```
<target name="init">
    <taskdef name="vfsmgen"
         classname="com.opencloud.sce.fsmtool.ant.VfsmGenTask">
         <classpath>
            <pathelement location="${basedir}/lib/state-machine-tool.jar" />
         </classpath>
    </taskdef>
</target>
```

Parámetros de la tarea Ant

La tarea Ant utiliza los siguientes atributos (todos son requeridos).

Tabla B1. Atributos de Ant para la VFSM

Atributo	Descripción
vfsmSpecificationFilename	Nombre del archivo de la especificación FSM a partir de la cual se va a generar el código. Por ejemplo: `FsmToolExampleStateMachineSbb.vfsm`
vfsmTemplateFilename	Ruta y nombre del archivo de la plantilla a utilizar para la generación del código (por ejemplo: `${basedir}/templates/vfsm.stg`, `${basedir}/templates/dot.stg`, `${basedir}/templates/vfsm-pojo.stg`)
package	Nombre del paquete para la salida
fileType	Sufijo para el archivo generado. Para `vfsm.stg` debe ser fijado en `java`. Para `dot.stg` debe ser fijado en `dot`.
class	Nombre de la clase para el código generado
superClass	La clase Java generada extiende de esta súper clase. Para `vfsm.stg` se requiere del nombre completo de la clase (no para `dot.stg`)
outputRootDirectory	El directorio raíz utilizado para escribir el archivo de la clase generada.

logPath	La ruta del log, para que sea fijada cuando se utilice el `vfsm-pojo.stg`. Para `vfsm.stg` se fija "".

Propiedades para las plantillas de generación de código

Los parámetros de la tarea Ant se guardan en archivos de propiedades, para utilizarlos en las plantillas de generación de código.

Otras tareas FSM

- ***Patrones o modelos FSM comunes***

Comprobación de los parámetros de un evento (Event-parameter checking)

Para comprobar los parámetros de un evento, usualmente las aplicaciones utilizan las acciones de entrada. Por ejemplo, cuando un servicio recibe un `InitialDP` o `SIP INVITE`, puede que necesite verificar que los parámetros del mensaje o los encabezados sean los correctos para la aplicación, antes de realizar la transición a otro estado para un futuro procesamiento. Se puede implementar este tipo de comprobación de parámetros definiendo una acción de entrada para el estado.

Manejo de errores

Hay dos maneras de manejar los errores en los métodos de acción:

- Fijar una entrada de error (*error input*) que haga que la máquina de estados realice la transición hacia un estado de error. Esto es lo cual es recomendado, ya que hace explícito el comportamiento del manejo de los errores en la especificación VFSM.
- Retornar un valor que no sea cero en el método de acción, esto permite el comportamiento por defecto o sobreescrito de `fsmTransitionActionError()`, en el cual se termina eficientemente el SBB. Esta manera se utiliza típicamente como un último recurso, donde (por alguna razón) no tiene sentido para la máquina de estados modelar el comportamiento.

Multiples llamadas al `SbbLocalObject`

Muchos servicios JSLEE constan de múltiples SBB, y se puede implementar sus FSM con la FSM Tool. Algunas de las FSM de los SBB resultantes recibirán entradas por medio de llamadas síncronas (llamadas de un SBB padre a un SBB hijo), a través de una interfaz `SbbLocalObject`. Si después de ejecutar un método de acción la FSM necesita un valor de retorno, ella fija la respuesta en un campo de instancia local, y lo envía de regreso después del retorno del método `receivedInput`. Este es el enfoque recomendado para la mayoría de las situaciones, no obstante también se pueden implementar llamadas de retorno y re-entrada para el SBB llamante, si es absolutamente necesario.

Los canales entre los componentes de JAIN SLEE pueden ser síncronos (utilizando las interfaces `SbbLocalObject` entre los SBBs) o asíncronos utilizando las Interfaces del Contexto de la Actividad (Activity Context interfaces, ACI).

Temporizadores

Las aplicaciones necesitan temporizadores cuando es posible que las redes no retornen una respuesta. Por ejemplo, a partir de la especificación de una VFSM, el siguiente código muestra un patrón en un estado para iniciar, cancelar o manejar un evento de temporizador:

```
state waitForMTForwardSM {
        entryaction sendMTForwardSM, startTimer;
        exitaction cancelTimer;
        transition if(MTForwardSMConf) nextState;
        transition if(Timer) failure;
      }
```

En este ejemplo, la FSM utiliza:

- La acción de ingreso `startTimer` para iniciar el temporizador de JSLEE
- La acción de salida `cancelTimer` para cancelar el temporizador, iniciado en `startTimer`

Si el temporizador expira, el SBB recibe un `TimerEvent`, y llama el método `receivedInput` con una entrada fija en `Timer`. Por ejemplo:

```
/**
     * @slee.event-method
     *   initial-event="False"
     *   event-type-name="javax.slee.facilities.TimerEvent"
     *   event-type-vendor="javax.slee"
     *   event-type-version="1.0"
     */
    public void onTimer(TimerEvent event, ActivityContextInterface aci) {

        this.receivedInput(Input.Timer, event, aci);
    }
```

Sintaxis del DSL de la VFSM

Se define una especificación VFSM con la palabra clave `vfsm`, seguida por el nombre de la VFSM, y luego un cuerpo que contiene `{` y `}`. Por ejemplo:

```
vfsm NameOfVfsmSpecification {

    ... vfsm specification body ...

}
```

- ***Cuerpo de la especificación VFSM***

El cuerpo de la especificación de la VFSM tiene los siguientes elementos:

➢ **Descripción**

Este elemento describe la VFSM, utilizando la palabra clave `description` seguida por el texto entrecomillado (el cual puede abarcar múltiples líneas).

```
description "text of the description of the vfsm";
```

- **Estado**

Este elemento define los estados en la VFSM, incluyendo las acciones y transiciones.

Palabra clave + nombre del estado + cuerpo

Se define un estado con la palabra clave `state`, seguida por el nombre del estado, y luego el cuerpo del estado.

```
state someState {
        ... state body ...
    }
```

Estado inicial y estado final

Se definen el estado inicial y final utilizando los modificadores `initial` y `final`. Por ejemplo, para un estado inicial:

```
initial state startState {
        ... state body ...
    }
```

Solamente se permite un estado inicial. En la versión actual de la FSM Tool el primer estado declarado debe tener el modificador `initial`.

- **Cuerpo del estado**

El orden de la declaración de los elementos debe ser el orden en que se listan en la Tabla B2, a continuación.

Tabla B2. Sintaxis del DSL del cuerpo del estado

Elemento	Palabra clave + …	Ejemplo	Número
description	`Description`	`description "text of the description of the state";`	
entry actions	`entryaction` + lista de acciones separadas por comas	`entryaction setTimer, sendMOForwardSM;`	0 o 1
exit actions	`exitaction` + lista de acciones		0 o 1

	separadas por comas	`exitaction cancelTimer;`	

input actions	`inputaction` + condición y lista de acciones separadas por comas	`inputaction if(hello) checkWeather;`
		`inputaction if(hello && niceWeather) checkTemperature;`
		• La primera línea en este ejemplo evalúa la expresión condicional `hello`; y si es verdadera, ejecuta la acción de salida `checkWeather`.
		• La segunda línea evalúa la expresión condicional `hello && niceWeather`; y si tanto la entrada `hello` como `niceWeather` son verdaderas, ejecuta la acción de salida `checkTemperature`.
		• La declaración `if()` debe contener una expresión condicional.
			0 o más

transitions	`transition` + condición y nombre del estado de destino	`transition if(hello) stopState;`
		`transition if(idp && notbarred) continueCall;`
		• La primera línea en este ejemplo evalúa la expresión condicional `hello`; si es verdadera, realiza la transición y realiza las acciones según el modelo de ejecución de la VFSM.
		• La segunda línea evalúa la expresión condicional `idp && notbarred`; si es verdadera, realiza la transición hacia el estado `continueCall`.
		`transition continueCall;`
		• Este ejemplo muestra una transición sin condición. En este estado, la máquina de estados nunca esperará una entrada ya que esta transición será realizada.
		• Solamente debería declararse una transición sin condición, y esta transición siempre debería ser la última. Cualquier transición declarada después de ella, sería inalcanzable, y nunca sería comprobada o realizada.
		• Para transiciones múltiples con la evaluación de verdadero de los condicionales, la primera transición declarada tomará la prioridad y será la única transición a ejecutarse.
		• La declaración `if()` debe contener una expresión condicional.
			0 o más

- **Diccionario de entrada**

Debe declarar todas las entradas utilizadas en los elementos del cuerpo del estado. Se define un elemento de diccionario de entrada con la palabra clave `inputdictionary`, seguida por el cuerpo del diccionario de entrada.

```
inputdictionary {
        ... inputdictionary body ...
    }
```

- **Cuerpo del diccionario de entrada**

El cuerpo del diccionario de entrada declara 0 o más entradas, cada una con un nombre de entrada seguido por un cuerpo encerrado entre `{` y `}`.

```
hello {
            ... input declaration body ...
      }
```

Cada cuerpo de declaración de la entrada puede contener los siguientes elementos:

- `description`: palabra clave seguida por el texto entrecomillado.
- `inputtype`: definido con la palabra clave `inputtype`, seguida ya sea por `transient` o `durable`; por defecto, sino está especificado, es `transient`.
- `virtualinputoroutput` – (*para uso futuro – no tiene influencia en esta versión*)

- **Diccionario de salida**

Debe declarar todas las acciones de ingreso (*entry*), salida (*exit*) y entrada (*input*) utilizadas en los elementos del cuerpo del estado. Un elemento de diccionario de salida está definido con la palabra clave `outputdictionary`, seguida por el cuerpo del diccionario de salida.

```
outputdictionary {
        ... outputdictionary body ...
    }
```

- **Cuerpo del diccionario de salida**

El cuerpo del diccionario de salida declara 0 o más acciones de salida, cada una con un nombre de acción, seguido por un cuerpo encerrado entre `{` y `}`.

```
setTimer {
      ... output declaration body ...
      }
```

Cada cuerpo de declaración de la salida puede contener los siguientes elementos:

- `description`: palabra clave seguida por el texto entrecomillado.
- `virtualinputoroutput`: (*para uso futuro – no tiene influencia en esta versión*)

B.3. ADAPTADOR DE RECURSOS SIP

JAIN SLEE interactúa y se comunica con los sistemas externos a través de la capa de RA. Todos los flujos de eventos que ingresan y salen del SLEE son transmitidos a través de la entidad RA adecuada. Todos los componentes software de JAIN SLEE, tales como componentes de biblioteca, lógica de los servicios, o implementaciones de un tipo de RA, entre otros, deben ser desplegados como Unidades Desplegables (Deployable Units, DU). Cada DU contiene las clases Java requeridas, junto con un descriptor de despliegue, obligatorio, en forma de archivo XML.

Los RA, SIP y SOAP, utilizados en MIDDIS proporcionan la abstracción a bajo nivel de la pila de protocolos y permiten la creación, modificación y el intercambio en dos vías de los mensajes del protocolo. El procedimiento general para proveer un RA particular requiere tres pasos:

1. **Despliegue del *tipo de RA (RA Type)***: contiene y define un conjunto de interfaces disponibles dentro del SLEE para interactuar con los recursos del RA.
2. **Despliegue de la *implementación del tipo de RA***: implementación particular de uno o más tipos de RA.
3. **Creación y activación de la *entidad del RA (RA Entity)***: la entidad es una instancia del RA, es decir, que por ejemplo una entidad específica del SIP RA escucha en el puerto IP específico.

Con el fin de permitir que una entidad de SBB se comunique a través de un RA específico, se necesita incluir en el descriptor de despliegue del SBB la configuración apropiada del enlace al tipo de RA particular.

El Open Cloud SIP RA Type API está basado en JAIN SIP, con algunas extensiones propietarias para las aplicaciones SLEE. Por defecto está incluido en el paquete del Rhino SDK. Sin embargo, se utilizó una versión del SIP Connectivity Pack más reciente: 2.2_06 (OpenCloud, 2010b).

El SIP RA proporciona la interfaz entre los UA SIP y el SLEE, y por tanto es el responsable del envío y la recepción de mensajes SIP a través de la red. El SIP RA procesa los mensajes provenientes de la red y los convierte en actividades y eventos SLEE para que sean utilizados por las aplicaciones del SLEE.

La configuración por defecto del SIP AR debería ser la adecuada para la mayoría de entornos, aunque se puede hacer la reconfiguración para un requerimiento en particular, como utilizar una dirección IP o un número de puerto diferentes. Es decir, que cada adaptador de recursos define un conjunto de propiedades de configuración por defecto, el cual determina su comportamiento. Por lo tanto, cuando un administrador crea una entidad de un RA (o sea una instancia del RA), puede especificar valores para las propiedades de configuración distintos a los que están por defecto.

B.3.1. Configurar el SIP RA

Propiedades de configuración por defecto

La configuración por defecto del SIP RA debe ser la adecuada para la mayoría de entornos. Sin embargo, se puede hacer la reconfiguración para un requerimiento en particular, como utilizar una dirección IP o un número de puerto diferentes. Cada adaptador de recursos define un conjunto de propiedades de configuración por defecto, el cual determina su comportamiento. Cuando un administrador crea una entidad de un RA (o sea una instancia del RA), puede especificar valores para las propiedades de configuración, distintos a los que están por defecto.

A continuación se presentan las instrucciones para la edición de las propiedades de configuración del SIP RA.

- ***Edición de las propiedades de configuración del SIP AR***

El script de construcción Ant `build.xml` crea la entidad del adaptador de recurso SIP con las siguientes propiedades (definidas en el archivo `build.properties`):

```
sip.ra.properties=IPAddress=AUTO,Transports="udp,tcp",Port=5060,
SecurePort=5061
```

Para cambiar estas propiedades, y especificar otras, editar la propiedad `sip.ra.properties`. Por ejemplo, para habilitar el soporte al *SIP RA's replicated-dialog*, se añadiría esta propiedad como sigue (y luego se despliega el SIP RA):

```
sip.ra.properties=IPAddress=AUTO,Transports="udp,tcp",Port=5060,
SecurePort=5061,ReplicatedDialogSupport=true
```

El script Ant `build.xml` pasará estas propiedades al comando de gestión `createRAEntity`.

B.3.2. Despliegue del SIP RA

Instalación de Ant

- Descargar Ant de http://ant.apache.org/bindownload.cgi. En este caso se descargo: `apache-ant-1.8.0-bin.tar.gz`.

- Ubicarse en el directorio donde se desee realizar la instalación, se recomienda `/usr/local/`, y realizar los siguientes pasos:

```
~/Descargas$ sudo cp -p apache-ant-1.8.0-bin.tar.gz /usr/local/
```

Descomprimir el archivo con el siguiente comando:

```
sudo tar -zxvf apache-ant-1.8.0-bin.tar.gz
```

- Una vez terminada la instalación se recomienda cambiar el nombre del `directorio apache-ant-1.8.0` a simplemente `ant`; quedando instalado en una ruta absoluta: `/usr/local/ant`

```
/usr/local$ sudo mv apache-ant-1.8.0 ant
```

- Ant requiere configurarse con diversas variables de entorno para su correcta operación:

- `ANT_HOME`: Indica el directorio raíz de instalación de Ant. De acuerdo a las instrucciones anteriores esta ruta sería : `/usr/local/ant`
- `PATH`: Define la ruta de acceso para los binarios del sistema; la modificación de esta variable permite acceder a los ejecutables de Ant (ant) desde cualquier directorio.

Las variables anteriores pueden ser definidas de dos maneras:

- *Nivel Global*: Permite que las variables estén accesibles a todo usuario del sistema, permitiendo que cualquier usuario utilice Ant; estas definiciones son colocadas en el archivo `/etc/profile` del sistema.
- *Nivel Usuario*: Las variables serían definidas únicamente para el usuario que utilice Ant; estas definiciones son colocadas en el archivo `~/.bashrc`, donde `~/` es el directorio base del usuario.

Independientemente de los métodos mencionados anteriormente, las declaraciones en estos archivos son idénticas:

```
sudo gedit /etc/bash.bashrc
```

En el archivo que se abre:

```
# Java Configuration

JAVA_HOME=/usr/lib/jvm/jdk1.6.0_20

export JAVA_HOME

ANT_HOME=/usr/local/ant

export ANT_HOME

PATH=${PATH}:${JAVA_HOME}/bin:${ANT_HOME}/bin
export PATH
```

Para verificar la correcta instalación de Ant realizar la siguiente prueba:

```
~$ echo $ANT_HOME
/usr/local/ant
~$ ant
Buildfile: build.xml does not exist!
Build failed
~$ ant -version
Apache Ant version 1.8.0 compiled on February 1 2010
```

Despliegue del SIP RA

Luego de configurar las propiedades por defecto, se puede desplegar el SIP RA en el SLEE. Configurar la IP del dominio en el archivo: `$RHINO_HOME/examples/sip-examples-2.2_06/sip.properties`, en la línea:

```
PROXY_DOMAINS=opencloud.com,opencloud.co.nz
```

El script `build.xml` de Ant contiene los targets para el despliegue y la anulación del despliegue del SIP RA, como sigue. Para desplegar el SIP RA, primero asegurarse que el SLEE está corriendo, y luego correr el `build.xml` con el target `deploysipra`, desde la ruta: `$RHINO_HOME/examples/sip-examples-2.2_06/`:

```
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant deploysipra
```

Este script instala la unidad desplegable del SIP RA y las librerías requeridas en el SLEE, y crea y activa la entidad del adaptador de recursos.

Anular el despliegue del SIP RA

Para anular el despliegue del SIP RA, correr `build.xml` con el target `undeploysipra`. Por ejemplo:

```
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant undeploysipra
```

Este script desactiva y remueve la unidad del adaptador de recursos y desinstala la unidad de despliegue del SIP RA.

B.3.3. Habilitar el Trazado de los Servicios SIP

Los servicios SIP pueden escribir la información de seguimiento en el sistema de registro del Rhino SLEE, utilizando la facilidad de trazado del SLEE. Esto es útil para entender lo que están haciendo los diferentes servicios.

Habilitar el trazado (o seguimiento) para un SBB particular

Para habilitar el trazado de la salida de un SBB en particular, utilizar el comando `setTraceLevel` en la consola de Rhino, como sigue:

```
~/rhino/RhinoSDK/client/bin$ sudo ./rhino-console
[sudo] password for ximena:
Interactive Rhino Management Shell
Rhino management console, enter 'help' for a list of commands
[Rhino@localhost (#0)]
```

Aquí, entre otras funciones, se pueden comprobar las unidades desplegables o los servicios desplegados, por ejemplo:

```
[Rhino@localhost (#0)] listdeployableunits
DeployableUnitID[url=file:/home/ximena/rhino/RhinoSDK/lib/javax-
slee-standard-types.jar]
[Rhino@localhost (#1)] listservices
There are no services installed
```

Por tanto, como ejemplo de habilitación del trazado, primero se despliega el SIP RA y los Servicios Proxy, Registrar, Presence.

```
~/rhino/RhinoSDK$ cd examples/
~/rhino/RhinoSDK/examples$ cd sip-examples-2.2_06
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant deploysipra
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant
deployregistrar
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant deployproxy
```

```
~/rhino/RhinoSDK/examples/sip-examples-2.2_06$ ant
deploypresence
```

Ahora se habilita el trazado para cada uno de estos servicios. Desde la consola cliente de Rhino se comprueban las unidades desplegadas y la lista de servicios:

```
[Rhino@localhost (#0)] listdeployableunits
DeployableUnitID[url=file:/home/ximena/rhino/RhinoSDK/lib/javax-slee-
standard-types.jar]
DeployableUnitID[url=file:jars/sip-ac-location-service.jar]
DeployableUnitID[url=file:jars/sip-presence-event.jar]
DeployableUnitID[url=file:jars/sip-presence-service.jar]
DeployableUnitID[url=file:jars/sip-proxy-service.jar]
DeployableUnitID[url=file:jars/sip-registrar-service.jar]
DeployableUnitID[url=file:lib/jsip-library-du-1.2.jar]
DeployableUnitID[url=file:lib/jsip-ratype-du-1.2.jar]
DeployableUnitID[url=file:lib/oc-javax-sip-library-du-2.1.jar]
DeployableUnitID[url=file:lib/ocjainsip-ra-du-2.2_06.jar]
DeployableUnitID[url=file:lib/ocsip-ra-type-du-2.2.jar]
[Rhino@localhost (#1)] listservices
ServiceID[name=SIP AC Location Service,vendor=OpenCloud,version=1.7]
ServiceID[name=SIP Notification Service,vendor=OpenCloud,version=1.1]
ServiceID[name=SIP Presence Service,vendor=OpenCloud,version=1.1]
ServiceID[name=SIP Proxy Service,vendor=OpenCloud,version=1.8]
ServiceID[name=SIP Publish Service,vendor=OpenCloud,version=1.0]
ServiceID[name=SIP Registrar Service,vendor=OpenCloud,version=1.8]
```

- ***Habilitación del trazado para el servicio SIP Proxy:***

```
[Rhino@localhost (#2)] settracerlevel sbb service=ServiceID[name=SIP\
Proxy\
Service,vendor=OpenCloud,version=1.8],sbb=SbbID[name=ProxySbb,vendor=O
penCloud,version=1.8] root finest
Set trace level of SbbNotification[service=ServiceID[name=SIP Proxy
Service,vendor=OpenCloud,version=1.8],sbb=SbbID[name=ProxySbb,vendor=O
penCloud,version=1.8]] root tracer to finest
```

- ***Habilitación del trazado para el servicio SIP Registrar:***

```
[Rhino@localhost (#3)] settracerlevel sbb service=ServiceID[name=SIP\
Registrar\
Service,vendor=OpenCloud,version=1.8],sbb=SbbID[name=RegistrarSbb,vend
or=OpenCloud,version=1.8] root finest
Set trace level of SbbNotification[service=ServiceID[name=SIP
Registrar
Service,vendor=OpenCloud,version=1.8],sbb=SbbID[name=RegistrarSbb,vend
or=OpenCloud,version=1.8]] root tracer to finest
```

- ***Habilitación del trazado para el servicio SIP Presence:***

```
[Rhino@localhost (#4)] settracerlevel sbb service=ServiceID[name=SIP\
Presence\
Service,vendor=OpenCloud,version=1.1],sbb=SbbID[name=EventStateComposi
torSbb,vendor=OpenCloud,version=1.0] root finest
Set trace level of SbbNotification[service=ServiceID[name=SIP Presence
Service,vendor=OpenCloud,version=1.1],sbb=SbbID[name=EventStateComposi
torSbb,vendor=OpenCloud,version=1.0]] root tracer to finest
```

B.4. ADAPTADOR DE RECURSOS SOAP

El Adaptador de Recursos SOAP del Rhino SLEE, que viene dentro del Web Connectivity Pack de Open Cloud, permite que servicios SLEE reciban como eventos las peticiones SOAP, y además permite que inicien peticiones SOAP hacia sistemas externos (OpenCloud, 2010c). Este paquete incluye: Binarios del SOAP y HTTP RA, Generador de carga HTTP, Servicios de ejemplo y código fuente, y los Javadoc respectivos.

El SOAP RA da soporte a SOAP sobre HTTP, como se encuentra especificado en la W3C SOAP 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/).

A continuación se muestran los pasos requeridos para desplegar en el SLEE el SOAP RA.

1- Iniciar el SLEE:

```
~/rhino/RhinoSDK$ sudo ./start-rhino.sh
```

2- Revisar la propiedad `listen.port` en el archivo `build.properties` del SOAP RA:

La propiedad `listen.port` especifica el puerto en el que el SOAP RA intentará escuchar cuando esté activo, y es el puerto con el que se comunicará el cliente de prueba. El puerto por defecto es `8000`.

```
# RA properties
listen.port=8090
```

3- Desplegar el SOAP RA. Una forma de desplegar el SOAP RA es mediante el despliegue de los ejemplos que vienen con el RA. Para esto se utiliza Ant, y se construye con el target "`deployexamples`". Esto desplegará el SOAP RA y los servicios SOAP de ejemplo.

```
~/rhino/RhinoSDK/examples/soap-2.1$ ant deployexamples
```

B.5. OPEN IMS CORE Y RHINO SLEE

B.5.1. Open IMS Core

1- Prerrequisitos

Para poder instalar el Open IMS Core (Fraunhofer FOKUS, 2009) es necesario instalar los siguientes paquetes en el sistema: Subversion, make, JDK1.5 o superior, ant, MySQL (*mysql-client-5.0 mysql-server-5.0)*, bison, flex, libxml2, libmysql ambos con los archivos development (*libmysqlclient15-dev, libxml2-dev)*, bind instalado y corriendo (*bind9*).

```
ximena@xvelasco:~$ sudo su
[sudo] password for ximena:
root@xvelasco:/home/ximena# add-apt-repository "deb
http://archive.canonical.com/ lucid partner"
```

Para descargar e instalar los anteriores paquetes, se puede utilizar un programa gestor de paquetes o a través de un terminal por medio del comando: *`sudo apt-get install`* `nombre del paquete`.

```
apt-get install ant mysql-client-5.0 mysql-server-5.0
libmysqlclient15-dev
apt-get install bind9 flex bison libxml2-dev subversion
apt-get install linux-headers-$(uname -r) ipsec-tools openssl
apt-get install libmysql++3 libmysql++-dev
```

2- Descargar el Open IMS Core

Para descargar los archivos fuente del Open IMS Core es necesario tener instalado el paquete Subversion.

Como primer paso se debe crear el directorio en donde se va a descargar los archivos fuente de la siguiente manera:

```
sudo mkdir /opt/OpenIMSCore/
```

A continuación se deben crear los directorios correspondientes para el CSCF y el HSS:

```
cd opt/OpenIMSCore
sudo mkdir ser_ims
sudo mkdir FHoSS
```

Una vez completados los pasos anteriores, se procede a descargar los archivos fuente por medio de los siguientes comandos:

```
sudo svn checkout
http://svn.berlios.de/svnroot/repos/openimscore/ser_ims/trunk
ser_ims
```

R/Revisión obtenida: 1014

```
sudo svn checkout
http://svn.berlios.de/svnroot/repos/openimscore/FHoSS/trunk
FHoSS
```

R/Revisión obtenida: 1014

Nota: Para llevar a cabo la descarga de los archivos fuente del Open IMS Core se debe tener acceso a Internet a través de una dirección IP real, de lo contrario no será posible la descarga.

Dado el caso de que se requiera descargar los archivos fuente en un equipo que tenga un acceso a Internet a través de un servidor Proxy, se puede proceder de la siguiente manera:

Descargar los siguientes archivos del servidor FTP ftp://ftp.berlios.de/pub/openimscore/snapshots/:

- ✓ `FhoSS2008XXX.r05XX.tgz`
- ✓ `ser_ims2008XXX.r05XX.tgz`

Llevar los archivos comprimidos al equipo en donde se quieren instalar, primero modificar los `.tgz` y renombrarlos como `ser_ims.tgz` y `FHoSS.tgz`, y descomprimir de la siguiente manera:

```
cd opt/OpenIMSCore/
sudo tar zxvf ser_ims.tgz
sudo tar zxvf FhoSS.tgz
```

3- Compilar Archivos Fuente

El primer paso es compilar los archivos fuente, correspondientes al CSCF, de la siguiente manera:

```
cd opt/OpenIMSCore/ser_ims
sudo make install-libs all
```

Nota: Si durante el proceso de compilado se genera algún error, esto indica que probablemente no se han instalado correctamente todos los prerrequisitos.

El siguiente paso es compilar los archivos fuente del HSS. Para esto se debe tener instalado el JDK1.5 o superior. Editar o crear la variable de entorno `JAVA_HOME` para el usuario que realiza la compilación. Una vez se ha constatado que la versión del JDK es la correcta, se procede a compilar los archivos fuente del HSS de la siguiente forma:

```
cd opt/OpenIMSCore/FHoSS
sudo ant compile
sudo ant deploy
```

De esta forma tanto el CSCF como el HSS estarán listos para funcionar.

4- Bases de datos MySQL

El siguiente paso consiste en crear las bases de datos que requiere el Open IMS Core para llevar a cabo un correcto funcionamiento. Para esto se requiere tener instalado el paquete MySQL en el sistema.

Para crear las bases de datos se procede de la siguiente forma:

```
mysql -u root -p -h localhost
</opt/OpenIMSCore/ser_ims/cfg/icscf.sql
Enter password:
mysql -u root -p -h localhost
</opt/OpenIMSCore/FHoSS/scripts/hss_db.sql
Enter password:
mysql -u root -p -h localhost
</opt/OpenIMSCore/FHoSS/scripts/userdata.sql
Enter password:
```

Ahora se debe constatar que las bases de datos han sido creadas de manera satisfactoria, de la siguiente manera:

```
mysql -u root -p -h localhost
Enter password:
show databases;
```

Una vez realizados los pasos anteriores, las bases de datos deben verse de la siguiente forma:

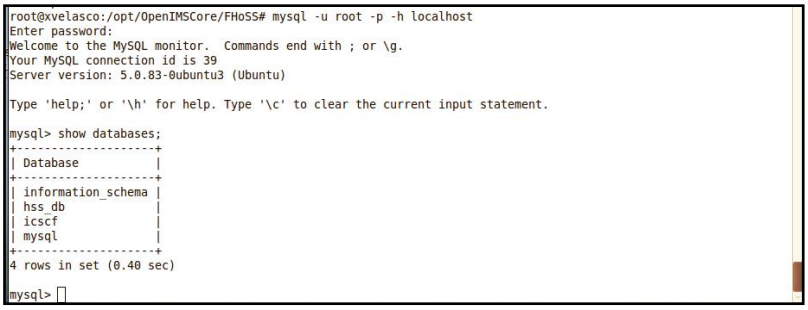


Figura B3. Bases de Datos del Open IMS Core

5- Configuración del servidor DNS

El servidor DNS que se instaló en el sistema, según los prerrequisitos, es el servidor *bind*. Este servidor DNS es el encargado de informar a cada uno de los componentes de la arquitectura del Open IMS Core en donde se encuentra cada componente y de esta manera permitir una comunicación exitosa entre ellos.

El procedimiento que se debe seguir para la configuración del servidor DNS es el siguiente:

- Copiar el archivo `open-ims.dsnzone` ubicado en `/opt/OpenIMSCore/ser_ims/cfg/` en la carpeta del servidor *bind* `/etc/bind` por medio del siguiente comando:

```
sudo cp /opt/OpenIMSCore/ser_ims/cfg/open-ims.dnszone /etc/bind/
```

- Antes de ejecutar los componentes se debe actualizar la fecha registrada en el archivo `open-ims.dnszone` ubicado en la carpeta `etc/bind`, como se muestra a continuación:

```
cd etc/bind
sudo gedit open-ims.dnszone
```

El archivo original tiene el siguiente contenido:

```
$ORIGIN open-ims.test.
$TTL 1W
@ 1D IN SOA localhost. root.localhost. (
2006101001 ; serial
3H ; refresh
15M ; retry
1W ; expiry
1D ) ; minimum

1D IN NS ns
ns 1D IN A 127.0.0.1

pcscf 1D IN A 127.0.0.1

open-ims.test. 1D IN A 127.0.0.1
icscf 1D IN A 127.0.0.1

_sip 1D SRV 0 0 5060 icscf
_sip._udp 1D SRV 0 0 5060 icscf
_sip._tcp 1D SRV 0 0 5060 icscf
```

```
open-ims.test. 1D IN NAPTR 10 50 "s" "SIP+D2U" "" _sip._udp
open-ims.test. 1D IN NAPTR 20 50 "s" "SIP+D2T" "" _sip._tcp


scscf 1D IN A 127.0.0.1

hss 1D IN A 127.0.0.1

ue 1D IN A 127.0.0.1

presence 1D IN A 127.0.0.1
```

El archivo debe quedar como se muestra en la Figura B4, a continuación:

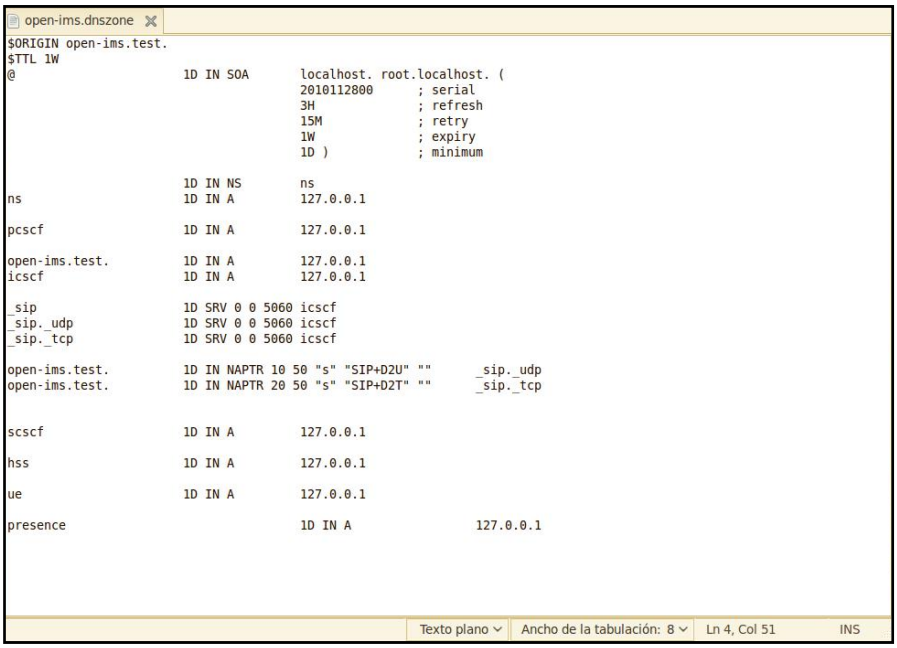


Figura B4. Archivo open-ims.dnszone

- Editar el archivo `named.conf.local`, ubicado en `/etc/bind/`, añadiéndole las siguientes líneas:

```
zone "open-ims.test" IN{
type master;
file "/etc/bind/open-
ims.dnszone";
notify no;
};
```

Para añadir las líneas anteriores al archivo `named.conf.local` se hace lo siguiente:

```
cd etc/bind
sudo gedit named.conf.local
```

Finalmente se guardan los cambios y se cierra la ventana. El archivo debe quedar como se muestra a continuación:

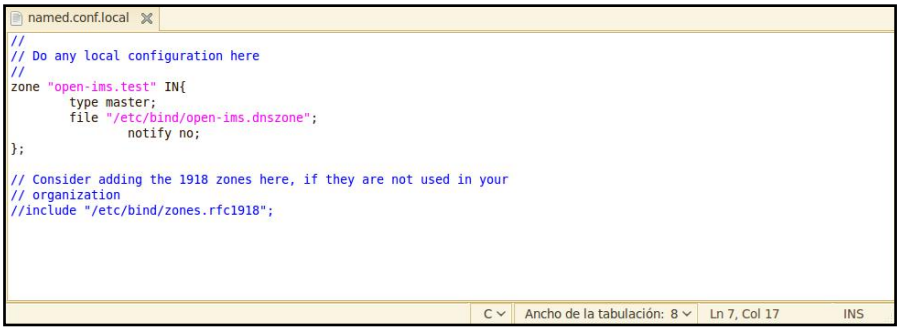


Figura B5. Archivo named.conf.local

- Reiniciar el servidor DNS de la siguiente manera: (**Nota**: cada vez que se reinicie el equipo se deben realizar los siguientes dos pasos)

  ```
  sudo /etc/init.d/bind9 restart
  ```

  Una vez reiniciado se debe ver como se muestra a continuación:

  ```
  root@xvelasco:/etc/bind# sudo /etc/init.d/bind9 restart
   * Stopping domain name service... bind9                    [ OK ]
   * Starting domain name service... bind9                    [ OK ]
  root@xvelasco:/etc/bind#
  ```

- Cuando se realiza algún cambio en la configuración de *bind*, es indispensable verificar los logs:

  ```
  # tail -n 50 /var/log/syslog
  ```

- Se confirma que el DNS carga la zona satisfactoriamente con el siguiente comando:

  ```
  #zone open-ims.test/IN: loaded serial
  Nov 28 12:29:48 xvelasco named[15259]: zone open-ims.test/IN:
  loaded serial 2010112800
  ```

- Se verifica el funcionamiento del DNS: `cd /etc/bind`

  ```
  # nslookup
  > server 127.0.0.1
  Default server: 127.0.0.1
  Address: 127.0.0.1 #53
  > open-ims.test
  Server: 127.0.0.1
  Address: 127.0.0.1 #53
  Name: open-ims.test
  Address: 127.0.0.1
  ```

- Se debe configurar la máquina en la cual se instaló el Open IMS Core con este DNS; para este fin, es posible usar el *applet Network-Manager*. Instalar el Network-Manager con paquetes de Synaptic, y reiniciar el equipo, esto actualiza el archivo `resolv.conf`. Al final se verifica el archivo `resolv.conf`.

  ```
  # gedit /etc/resolv.conf &
  ```

  El archivo original tiene el siguiente contenido:

```
# Generated by NetworkManager
domain mshome.net
search mshome.net
nameserver 192.168.0.1
```

- Editar el archivo `named.conf.options` ubicado en `/etc/bind/` agregándole las siguientes líneas:

```
forward first;
forwarders{
      dirección IP del servidor
      DNS de la LAN
};
```

Por lo tanto el archivo debe quedar como se muestra en la Figura B6:

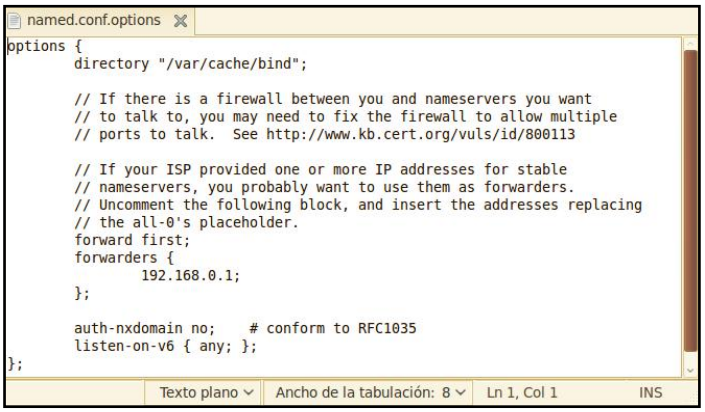


Figura B6. Archivo named.conf.options

En este caso, 192.168.0.1 es la dirección IP del servidor DNS de la LAN. La dirección IP del servidor DNS se la LAN se puede obtener en el archivo `resolv.conf` ubicado en `/etc/` en la primera línea de `nameservers`.

A continuación ejecutar:

```
# cd /opt/OpenIMSCore/
# chown -R bind:bind /etc/bind/open-ims.dnszone
```

- Reiniciar el servidor DNS de la siguiente manera:

```
sudo /etc/init.d/bind9 restart
```

Cuando se realiza algún cambio en la configuración de bind, es indispensable verificar los logs:

```
# tail -n 50 /var/log/syslog
```

- A continuación ejecutar:

```
root@xvelasco:/etc# dig @127.0.0.1 pcscf.open-ims.test
```

Del resultado generado tener en cuenta la línea:

```
;; ANSWER SECTION:
pcscf.open-ims.test.    86400 IN    A     127.0.0.1
```

Entonces modificar el archivo de la siguiente manera:

```
root@xvelasco:/etc# gedit /etc/resolv.conf &
```

```
# Generated by NetworkManager
domain open-ims.test
search open-ims.test
nameserver 127.0.0.1
```

Realizar los siguientes pasos, toda vez que se reinicie el equipo:

```
sudo /etc/init.d/bind9 restart
tail -n 50 /var/log/syslog
```

- Finalmente, para comprobar que se ha realizado una configuración exitosa del servidor DNS, se hace ping a alguno de los componentes del Open IMS Core de la siguiente manera:

```
# cd /opt/OpenIMSCore/
# ping pcscf.open-ims.test
ping pcscf.open-ims.test
```

6- Ejecutar componentes

En este punto todos los componentes del sistema están correctamente configurados y por lo tanto, están listos para ser ejecutados. Para iniciar el sistema se debe inicializar cada uno de los componentes (P-CSCF, I-CSCF, S-CSCF y HSS) de la siguiente manera:

- Se deben copiar los archivos `pcscf.cfg`, `icscf.cfg`, `scscf.cfg`, `pcscf.xml`, `icscf.xml`, `scscf.xml`, `pcscf.sh`, `icscf.sh`, `scscf.sh` desde la carpeta `opt/OpenIMSCore/ser_ims/cfg` hacia la carpeta `opt/OpenIMSCore`.

```
# cd /opt/OpenIMSCore/
# cp ser_ims/cfg/*.cfg .
# cp ser_ims/cfg/*.xml .
# cp ser_ims/cfg/*.sh .
```

- Para ejecutar los diferentes componentes pertenecientes al CSCF se procede de la siguiente manera (**Nota**: tener en cuenta que todos los CSCF deben estar ejecutándose a la vez, y su orden de inicio es: P-CSCF, I-CSCF y S-CSCF, así mismo el HSS también debe estar ejecutándose):

```
# sudo su
# cd opt/OpenIMSCore
# ./pcscf.sh
# ./icscf.sh
# ./scscf.sh
```

- Finalmente se lanza el HSS de la siguiente manera:

```
# sudo su
# cd /opt/OpenIMSCore/FHoSS/deploy
# ./startup.sh
```

- Probar la ejecución de los componentes a través de la Consola Web de gestión del FHoSS. Los datos por defecto para el ingreso son: *Nombre de usuario*: `hssAdmin`,

Password: `hss`. No se recomienda dejar el puerto 8080, para ello debe configurarse el archivo:

```
# cd /opt/OpenIMSCore/FHoSS/deploy/
# gedit hss.properties &
```

Buscar: `port=8080` y cambiar por: `port=8089`, por ejemplo. Por lo tanto: al ejecutar en el navegador: `http://localhost:8089/hss.web.console/`, se obtiene la interfaz mostrada en la Figura B7:

Figura B7. Consola Web del FHoSS

B.5.2. Configuración del Rhino SLEE como servidor de aplicaciones en el OpenIMSCore

En la Consola Web del FHoSS realizar los procedimientos que se describen a continuación:

- En la pestaña: *Services* → *Application Servers* → *create*, crear el AS de Rhino SLEE como se muestra en la Figura B8, y al finalizar clic en *Save*.

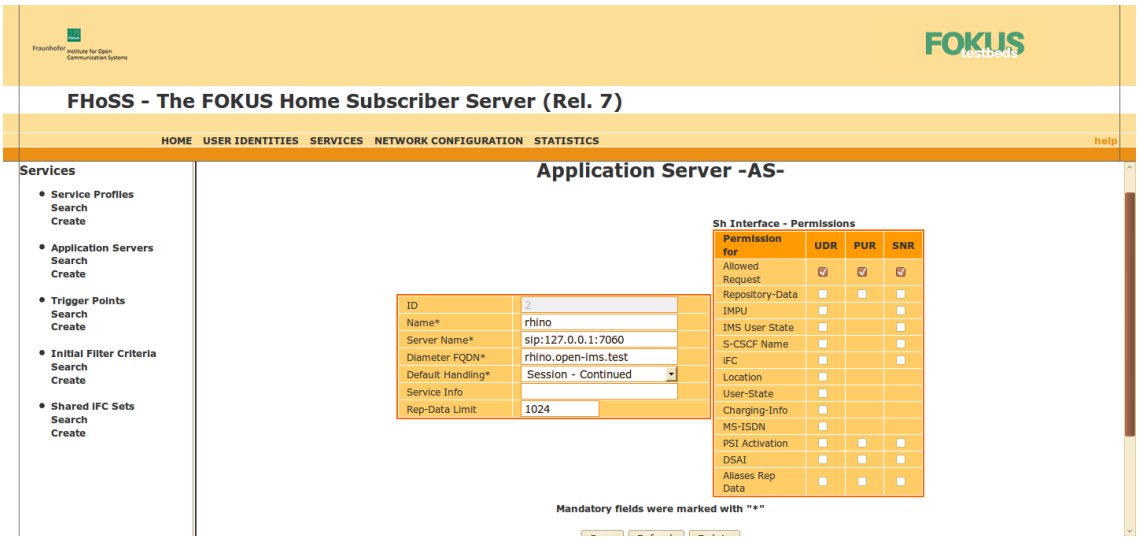


Figura B8. Creación de Rhino SLEE como Servidor de Aplicaciones

- En la pestaña: *Services* → *Trigger Points* → *create*, crear el *Trigger Point* para Rhino SLEE como se muestra en la Figura B9, y al finalizar clic en *Save*.

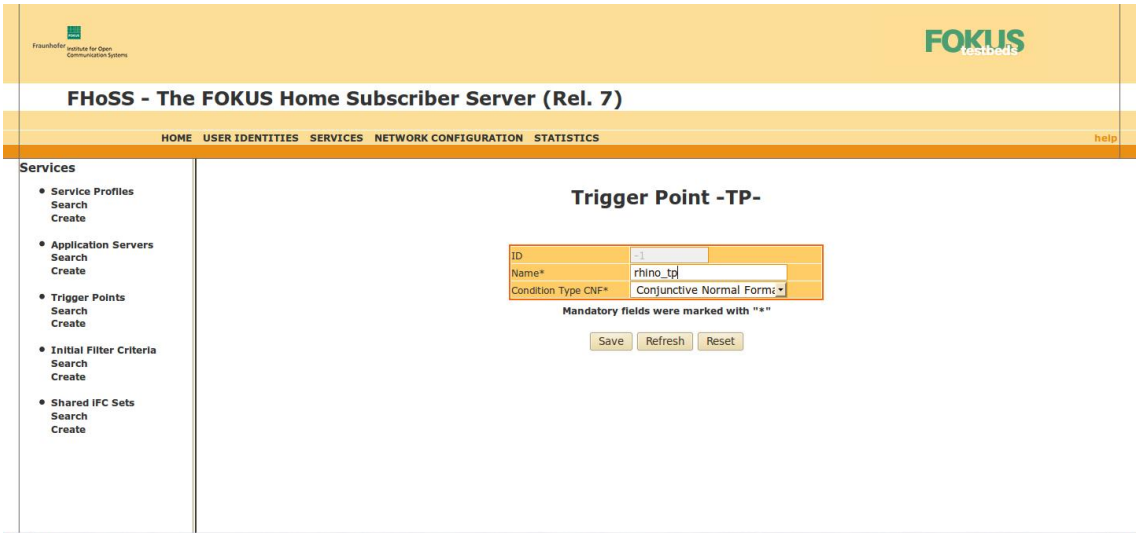


Figura B9. Creación del Trigger Point para Rhino SLEE

A continuación, la interfaz es extendida para realizar configuraciones adicionales que corresponden a los métodos SIP que se van a asignar al Trigger Point:

- Clic en el combo *Request-URI → Method → REGISTER*
- Clic en el combo *Request-URI → Method → INVITE*
- …, al finalizar clic en *Save*.

El resultado se muestra en la Figura B10:

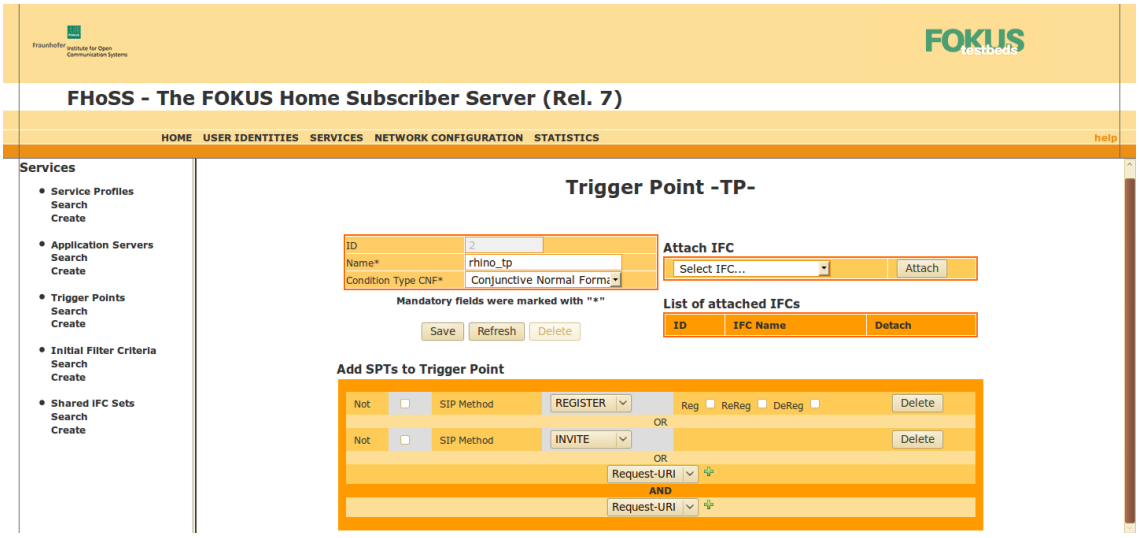


Figura B10. Adición de métodos SIP al Trigger Point para Rhino SLEE

- En la pestaña: *Services → Initial Filter Criteria → create*, crear el *Initial Filter Criteria* para Rhino SLEE como se muestra en la Figura B11, y al finalizar clic en *Save*.

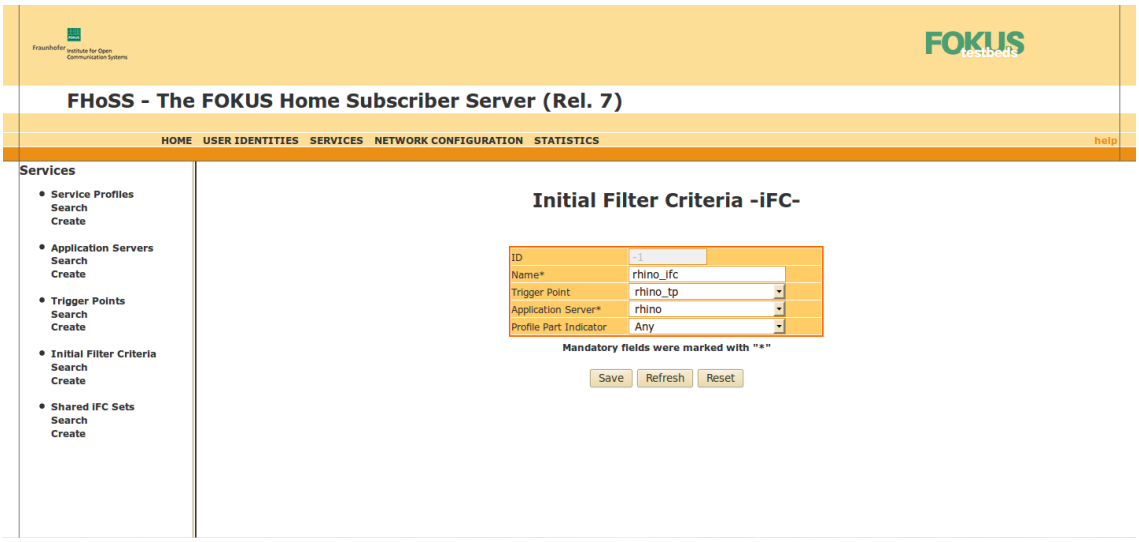


Figura B11. Creación del Initial Filter Criteria para Rhino SLEE

- En la pestaña: *Services* → *Service Profiles* → *create*, crear el *Service Profile* para Rhino SLEE como se muestra en la Figura B12, y al finalizar clic en *Save*.

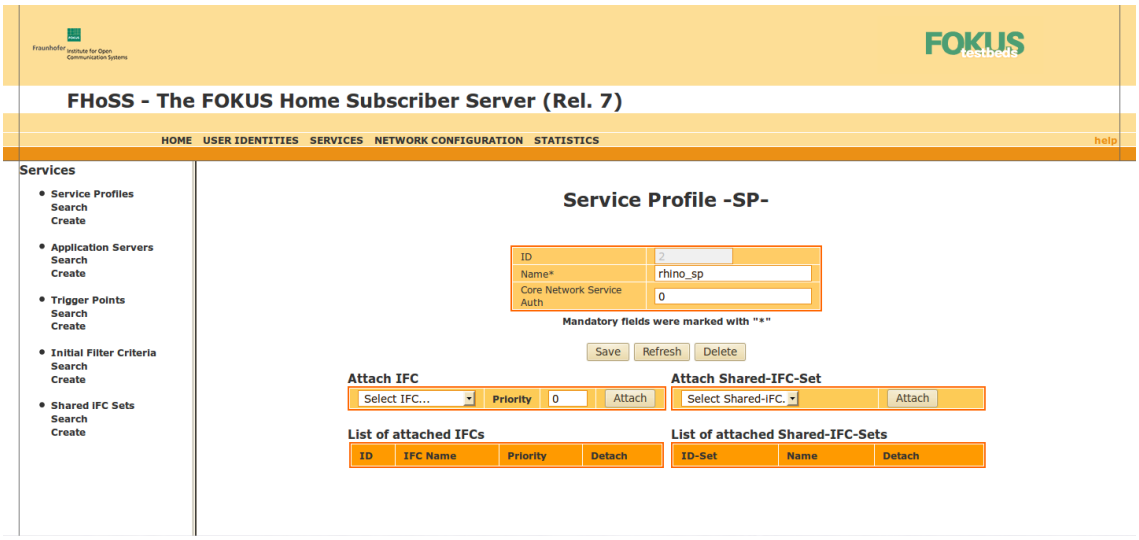


Figura B12. Creación del Service Profile para Rhino SLEE

A continuación, la interfaz es extendida y se despliega la opción *Attach IFC*, donde se selecciona el IFC creado previamente. Al hacer clic en *Save*, el resultado de este proceso se muestra en la Figura B13:

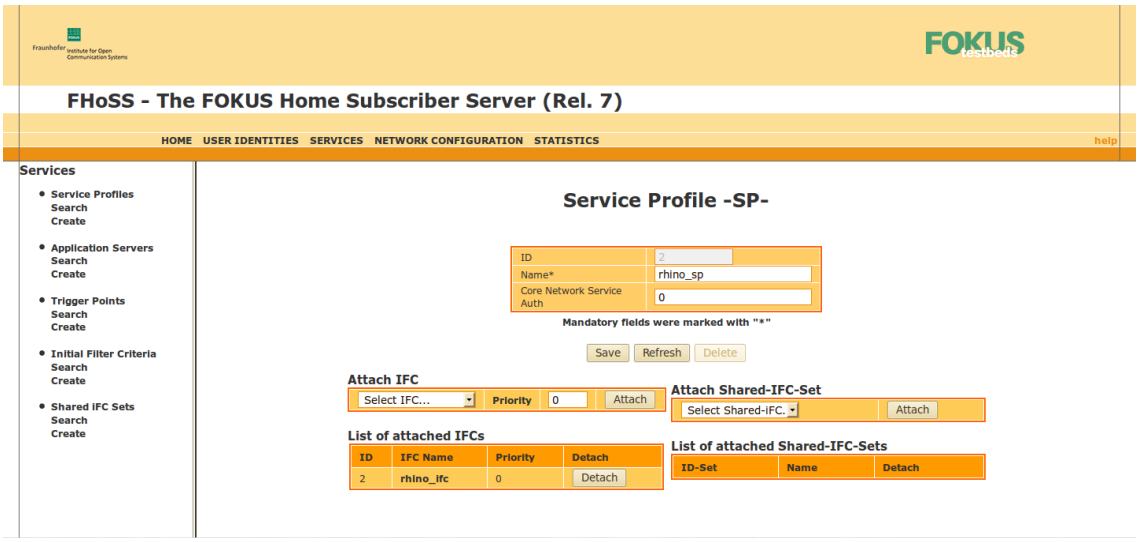


Figura B13. Configuración del Service Profile de acuerdo al IFC creado para Rhino SLEE

- En la pestaña: *User Identities* → *Public User Identity* → *search* → *search*, Figura B14,

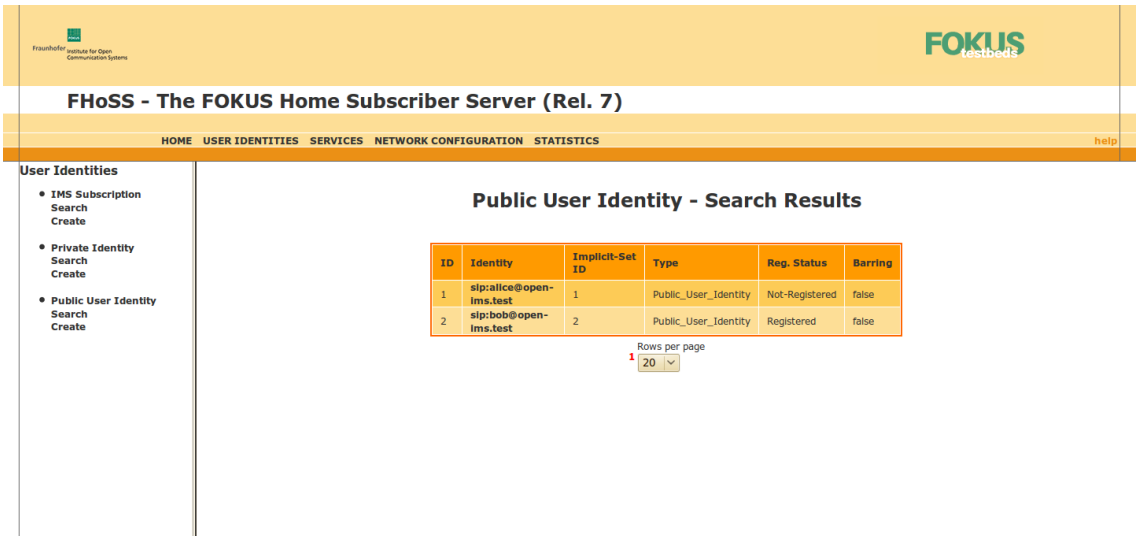


Figura B14. Pestaña de búsqueda de User Identities

Hacer clic en `alice@open-ims.test` y `bob@open-ims.test`, y seleccionar para cada uno el *Service Profile*: `rhino_sp`, como se muestra en la Figura B15, al finalizar clic en *Save*:

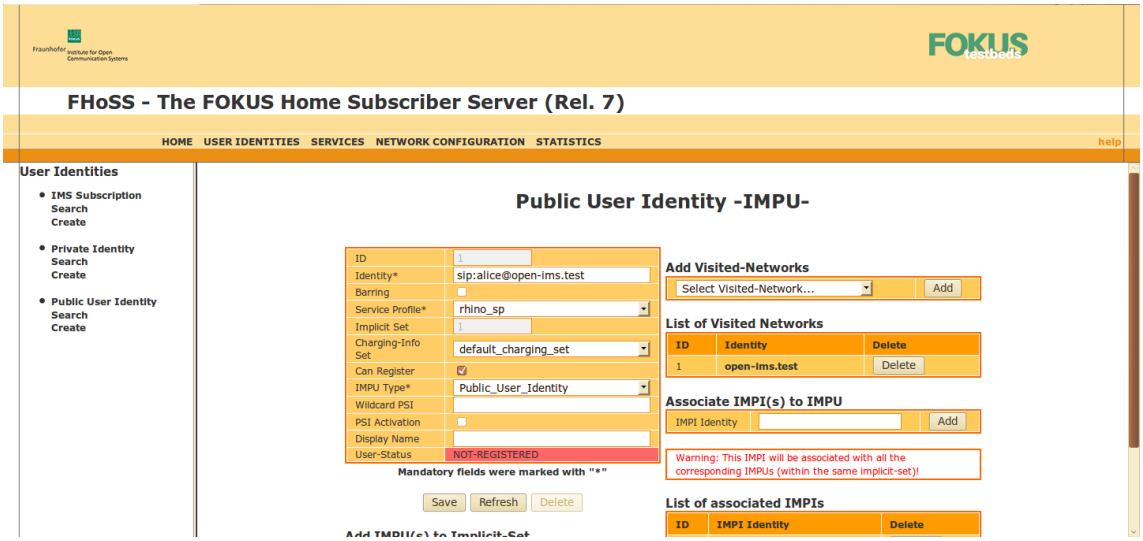


Figura B15. Configuración del Service Profile de Rhino SLEE para un Public User Identity

Para el caso de pruebas de MIDDIS se utilizaron los usuarios que el OpenIMSCore trae configurados por defecto. Específicamente, por una parte el usuario con dirección: `sip:bob@open-ims.test:7065`, representa al WS dentro de la red IMS, por otra, el usuario con dirección `sip:alice@open-ims.test:5070`, representa al Cliente IMS.

B.6. DISEÑO E IMPLEMENTACIÓN DE MIDDIS

B.6.1. Subsistema de Interfaces de Recursos de Red

La implementación del SSIRR de MIDDIS se realizó a través de un elemento constitutivo de la arquitectura de JSLEE: el plano de RA, debido a la correspondencia en características y funcionalidades requeridas e implementadas entre estos elementos. JSLEE proporciona capacidades de integración utilizando una arquitectura de plugins, conocida como la arquitectura del RA, la cual permite la interconexión de JSLEE con el mundo exterior; por ejemplo, proporciona las interfaces para las pilas de protocolos de comunicaciones, servicios de directorio o sistemas externos (tales como el rating y la facturación o *billing*).

B.6.2. Subsistema de Transmisión de Eventos

La implementación del SSTE de MIDDIS se realizó a través de un elemento que hace parte del marco de trabajo de JSLEE: el Enrutador de Eventos, debido también a la correspondencia en características y funcionalidades requeridas e implementadas entre estos elementos. El entorno de JSLEE invoca los SBB de acuerdo a un modelo estandarizado del ciclo de vida, similar al de los EJB. Durante las transacciones el entorno de ejecución asegura y gestiona el procesamiento de eventos y la invocación del marco de trabajo; al hacerlo, el servidor de aplicaciones de JSLEE permanece en un estado definido y consistente, incluso en caso de fallos.

B.6.3. Subsistema de Mediación IMS/SOA y Subsistema de Registro de Servicios IMS-WS

Para la creación de la lógica de servicio de MIDDIS, conformada por los subsistemas restantes: SSMIDD y SSREGS, se utilizó la FSM Tool, versión 0.8.5 (OpenCloud, 2009), en conjunto con

XDoclet versión 2.1.0, ambas de OpenCloud, las cuales son herramientas livianas, orientadas a simplificar la creación de servicios para Rhino SLEE.

El modelamiento de servicios a partir de FSM inicia con el diseño de los diagramas de secuencia para todos los procesos que la aplicación debe ejecutar. Una vez se tienen los diagramas de secuencia se puede empezar a crear un conjunto de máquinas de estados que manejen los flujos de llamada definidos en los diagramas. Se puede pensar en cada diagrama de secuencia como un camino a través de la máquina de estados.

Las FSM de los servicios que proporciona MIDDIS se definieron formalmente en un Lenguaje de Dominio Específico (Domain-Specific Language, DSL) basado en texto. Los principales pasos seguidos para el desarrollo de cada uno de los servicios de MIDDIS por medio de la FSM Tool fueron:

1. Se escribió la especificación de cada FSM, de manera que describiera completamente el comportamiento de los protocolos de comunicaciones SIP y SOAP. El modelado FSM de un protocolo de comunicaciones puede ser descrito como una tupla (Σ, Γ, S, s_0, δ, ω), donde Σ es el alfabeto de entrada, Γ es el alfabeto de salida, S es un conjunto no vacio y finito de estados, s_0 es el estado inicial (pertenece a S), δ es la función de la transición δ: S $\times$ Σ -> S, ω es la función de salida ω: S $\times$ Σ -> Γ (Basicevic, y otros, 2010).

   La especificación de una FSM consiste en un archivo de texto en el que se definen los estados, las acciones, las transiciones, las entradas y las salidas. La sintaxis del VFSM DSL incluye:

   - Definiciones y descripciones de los estados
   - Entradas, salidas, y acciones de ingreso asociadas con cada estado junto con las condiciones que se deben cumplir antes de la ejecución
   - Transiciones, con las condiciones que se deben cumplir antes de realizar la transición
   - Un diccionario de entrada (`inputdictionary`), en el que se declaran todas las entradas (eventos de entrada) a las que hacen referencia las definiciones de los estados
   - Un diccionario de salida (`outputdictionary`), en el que se declaran todas las salidas a las que hacen referencia las definiciones de los estados.

   De acuerdo a los servicios proporcionados por el SSMIDD y el SSREGS, y específicamente a los flujos de eventos específicos de cada protocolo de comunicación en los que basan su funcionalidad, se definieron 4 FSM, 2 para cada protocolo de comunicaciones dependiendo del tipo de servicios provistos.

   El primer grupo de especificaciones de FSM da soporte a los procesos de Registro de un WS en IMS, y a la Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI. Aportes fundamentales del presente trabajo.

- **SoapMiddStateMachine.vfsm**

La máquina de estados incluye (Figura B16):

- *Estados*: `INITIAL`, `REGISTERED`, `ERROR`, y `FINAL`

- *Eventos*: `soapRequest`, `register`, `successful`, `proccesing` y `error`
- Una acción de entrada para el estado `INITIAL`: `checkSoapRequest`
- Una acción de entrada para el estado `REGISTERED`: `checkSoapRequest`

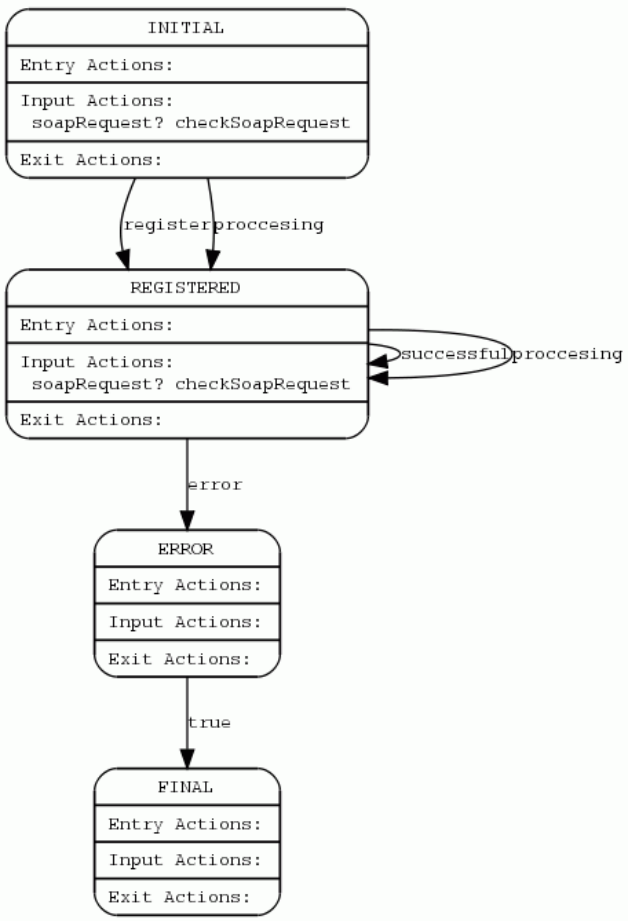


Figura B16. SoapMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `soapRequest`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapRequest` en la cual se analiza la solicitud SOAP recibida.
- Si se fijan las entradas `register` o `proccesing`, la máquina ejecuta la transición hacia el estado `REGISTERED`. Las entradas `register` y `proccesing` son fijadas para identificar los procesos de registro de un WS en IMS y de gestión UDDI en MIDDIS, respectivamente.

En el estado `REGISTERED`:

- Cuando la máquina recibe el evento `soapRequest`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapRequest` en la cual se analiza la solicitud SOAP recibida.
- La máquina ejecuta la transición hacia el mismo estado y hacia el estado ERROR al fijarse las entradas `successful|proccesing` y `error`, respectivamente. Las entradas `successful` y `proccesing` son fijadas para identificar los procesos de envío de

respuesta SOAP exitosa, desde el SLEE hacia la red subyacente, y de gestión UDDI en MIDDIS, respectivamente. La entrada `error` es fijada cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

Cuando la máquina de estados llega al estado `ERROR`, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado `FINAL`.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

- **SipMiddStateMachine.vfsm**

La máquina de estados incluye (Figura B17):

- *Estados*: `INITIAL`, `REGISTER`, `ERROR`, y `FINAL`
- *Eventos*: `clientErrorResponse`, `timeOut`, `registering`, `error`, `okResponse`, `successRegister` y `endSession`
- Acciones de entrada para el estado `INITIAL`: `clientErrorResponse`, `timeOut`
- Acciones de entrada para el estado `REGISTER`: `okResponse`, `timeOut`

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `clientErrorResponse`, debido a la llegada de un evento de respuesta SIP de tipo error del cliente o `401 UNAUTHORIZED` al JSLEE, ésta ejecuta la acción de entrada `checkClientError` en la cual se analiza la respuesta SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `registering` y `error`, la máquina ejecuta la transición hacia el estado `REGISTER` y `ERROR`, respectivamente. Las entradas `registering` y `error` son fijadas para identificar los procesos de generación de una petición SIP de registro hacia IMS y de ocurrencia de un error, respectivamente.

En el estado `REGISTER`:

- Cuando la máquina recibe el evento `okResponse`, debido a la llegada de un evento de respuesta SIP de tipo `200 OK` al JSLEE, ésta ejecuta la acción de entrada `checkOkRegister` en la cual se procesa la confirmación exitosa a la petición de registro del WS en IMS.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- La máquina ejecuta la transición hacia el mismo estado, hacia el estado `ERROR`, y hacia el estado `FINAL`, al fijarse las entradas `successRegister`, `error`, y `endSession`, respectivamente. La entrada `successRegister` se fija para identificar el proceso de confirmación de registro exitoso de un WS en IMS; la entrada `error` es fijada cuando se produce algún error durante uno de los procesos que ejecuta la lógica del servicio; y la entrada `endSession` se fija cuando se requiere terminar la ejecución de la máquina.

Cuando la máquina de estados llega al estado ERROR, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado FINAL.

En el estadoFINAL la máquina termina su ejecución y retorna al estado INITIAL.

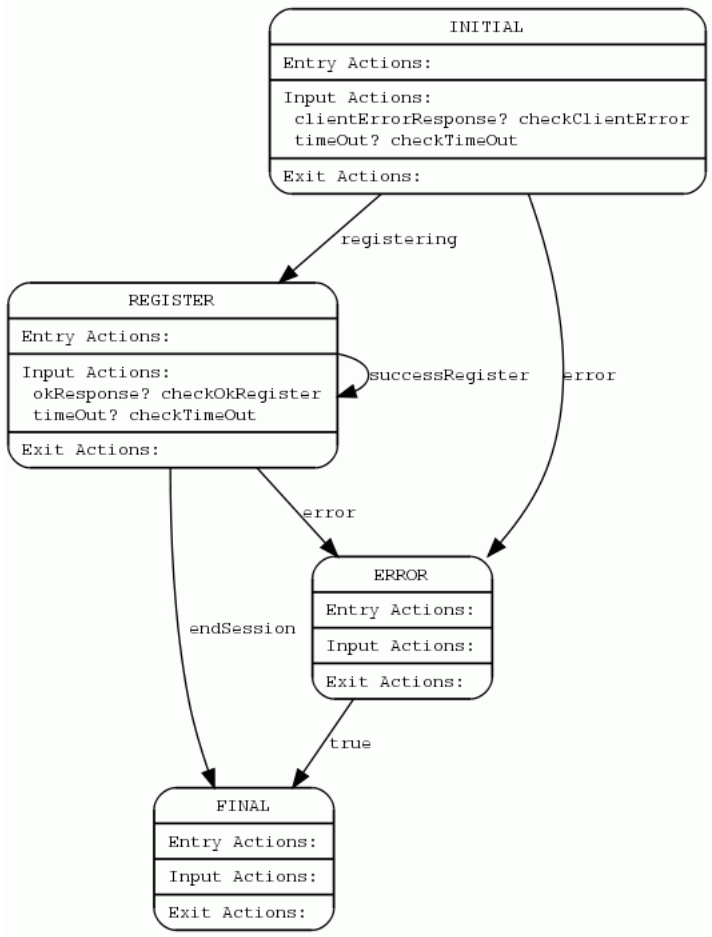


Figura B17. SipMiddStateMachine

El segundo grupo de especificaciones de FSM da soporte a los procesos de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS. Aportes fundamentales del presente trabajo.

- **SipUASMiddStateMachine.vfsm**

La máquina de estados incluye (Figura B18):

- *Estados*: INITIAL, CONNECTING, CONVERGED, ERROR, y FINAL
- *Eventos*: soapRequest, register, successful, proccesing y error
- Las acciones de entrada para el estado INITIAL: inviteRequest, timeOut, messageRequest, byeRequest
- Las acciones de entrada para el estado CONNECTING: ackRequest, timeOut
- Una acción de entrada para el estado CONVERGED: timeOut

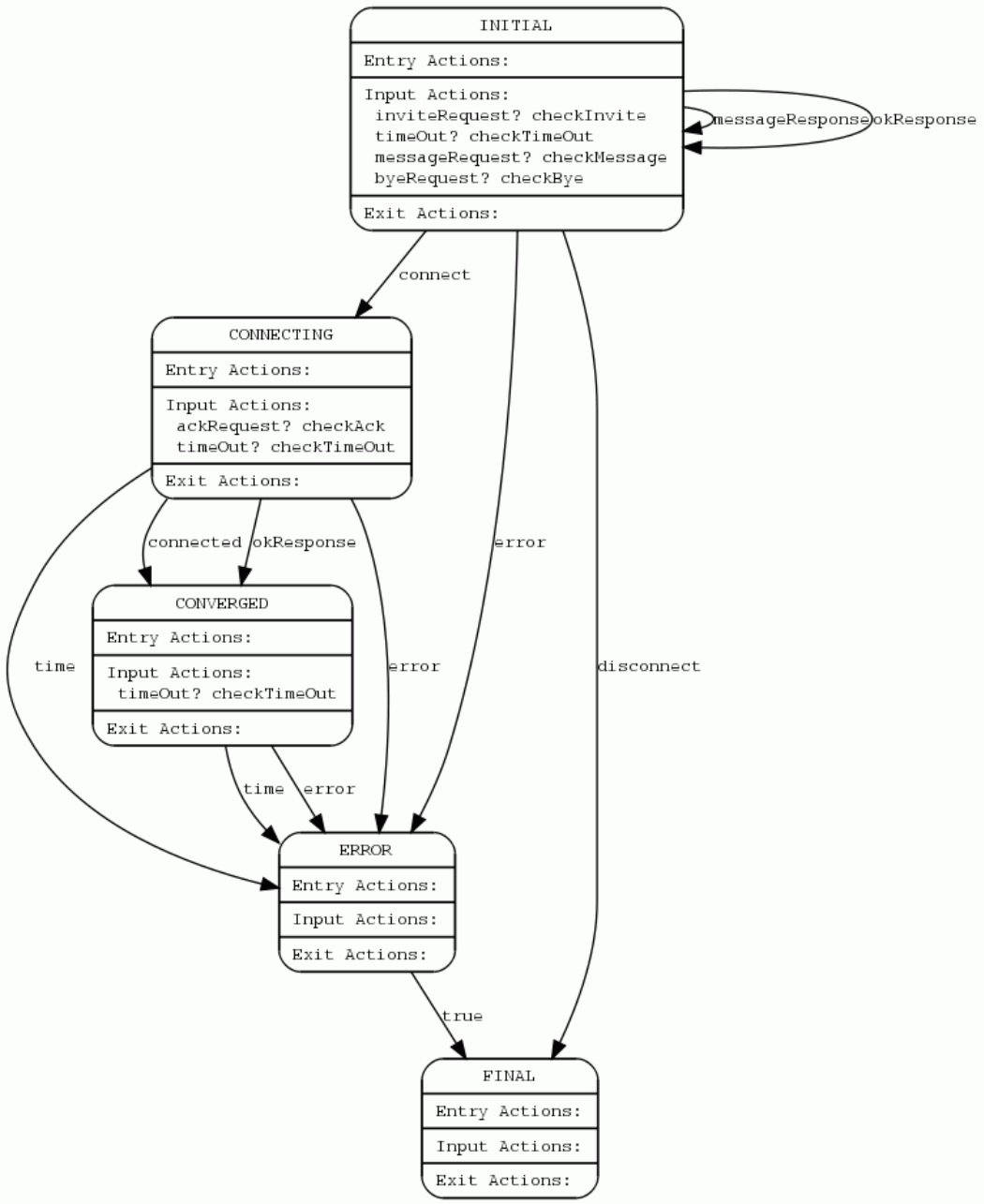


Figura B18. SipUASMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `inviteRequest`, debido a la llegada de un evento SIP de tipo `INVITE` al JSLEE, ésta ejecuta la acción de entrada `checkInvite` en la cual se analiza la petición SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Cuando la máquina recibe el evento `messageRequest`, debido a la llegada de un evento SIP de tipo `MESSAGE` al JSLEE, ésta ejecuta la acción de entrada `checkMessage` en la cual se procesa la invocación de un WS desde IMS.

- Cuando la máquina recibe el evento `byeRequest`, debido a la llegada de un evento SIP de tipo `BYE` al JSLEE, ésta ejecuta la acción de entrada `checkBye` en la cual se procesa la petición de finalización de sesión entre el UA IMS y el WS.
- Si se fijan las entradas `messageResponse`o`okResponse`, la máquina ejecuta la transición hacia el estado `INITIAL`; si se fijan las entradas `connect`, `disconnect`, y `error`, la máquina ejecuta la transición hacia los estados `CONNECTING`, `FINAL`, y `ERROR`, respectivamente. La entrada `messageResponse` se fija para identificar el proceso de generación de una petición SIP de tipo `MESSAGE` que contenga el resultado de la invocación al WS; la entrada `okResponse` se fija para identificar el proceso de recepción de una respuesta SIP de tipo `200 OK`; la entrada `connect` se fija para identificar que se ha iniciado el proceso de establecimiento de una sesión entre el UA IMS y el WS; la entrada `disconnect` se fija para identificar el proceso de finalización de una sesión entre el UA IMS y el WS; la entrada `error` se fija cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

En el estado `CONNECTING`:

- Cuando la máquina recibe el evento `ackRequest`, debido a la llegada de un evento SIP de tipo `ACK` al JSLEE, ésta ejecuta la acción de entrada `checkAck` en la cual se analiza la petición SIP recibida.
- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `connected` o `okResponse`, la máquina ejecuta la transición hacia el estado `CONVERGED`; si se fijan las entradas `time` o `error`, la máquina ejecuta la transición hacia el estado `ERROR`. La entrada `connected` se fija para identificar el proceso confirmación del establecimiento exitoso de sesión entre el UA IMS y el WS; la entrada `okResponse` se fija para identificar el proceso de envío de una respuesta SIP de tipo `200 OK`; la entrada `time` se fija para identificar la ocurrencia de un evento de temporizador durante alguno de los procesos que ejecuta la lógica del servicio; la entrada `error` se fija cuando se produce algún error durante alguno de los procesos que ejecuta la lógica del servicio.

En el estado `CONVERGED`:

- Cuando la máquina recibe el evento `timeOut`, debido a la llegada de un evento de temporizador, ésta ejecuta la acción de entrada `checkTimeOut` en la cual se fija la entrada `error`.
- Si se fijan las entradas `error` o `time`, la máquina ejecuta la transición hacia el estado `ERROR`. Las entradas `time` y `error` se fijan para identificar la ocurrencia de un evento de temporizador y de un error, respectivamente, durante alguno de los procesos que ejecuta la lógica del servicio.

Cuando la máquina de estados llega al estado `ERROR`, por causa de algún error en la ejecución de la lógica del servicio, ésta ejecuta inmediatamente la transición hacia el estado `FINAL`.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

- **SoapUASMiddStateMachine.vfsm**

La máquina de estados incluye (Figura B19):

- *Estados*: `INITIAL`, `CONVERGED`, y `FINAL`
- *Eventos*: `soapResponse`, `generatingSession`, `soapOkInviteResponse`, `invoking`y`endSession`
- Una acción de entrada para el estado `INITIAL`: `soapResponse`
- Una acción de entrada para el estado `CONVERGED`: `soapResponse`

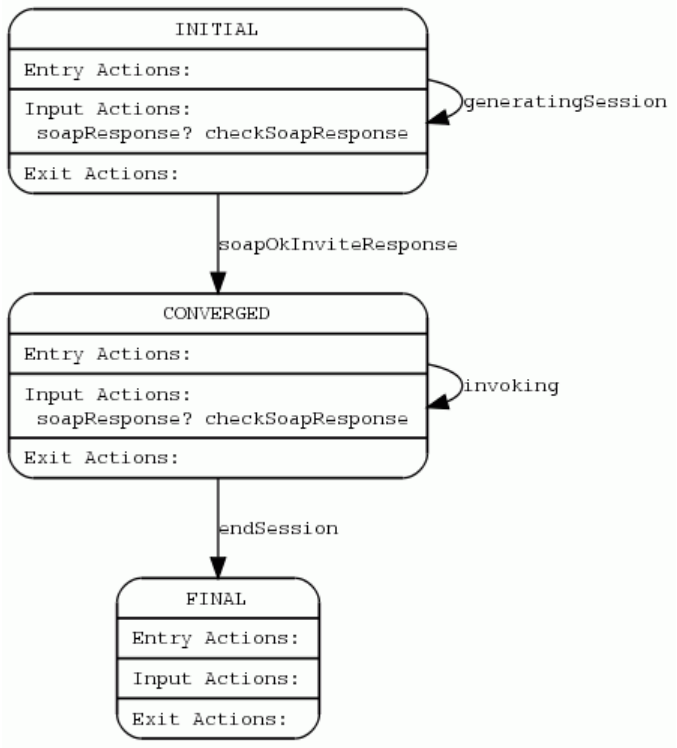


Figura B19. SipUASMiddStateMachine

En el estado `INITIAL`:

- Cuando la máquina recibe el evento `soapResponse`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapResponse` en la cual se procesa la respuesta SOAP recibida.
- Si se fijan las entradas `generatingSession` y `soapOkInviteResponse` la máquina ejecuta la transición hacia el mismo estado y hacia el estado `CONVERGED`, respectivamente. Las entradas `generatingSession` y `soapOkInviteResponse` son fijadas para identificar los procesos de generación de una petición de inicio de sesión desde el SLEE hacia el WS y de confirmación exitosa al inicio de sesión entre el UA IMS y el WS, respectivamente.

En el estado `CONVERGED`:

- Cuando la máquina recibe el evento `soapResponse`, debido a la llegada de un evento SOAP al JSLEE, ésta ejecuta la acción de entrada `checkSoapResponse` en la cual se procesa la respuesta SOAP recibida.

- La máquina ejecuta la transición hacia el mismo estado y hacia el estado `FINAL` al fijarse las entradas `invoking` y `endSession`, respectivamente. Las entradas `invoking` y `endSession` son fijadas para identificar los procesos de envío de una petición SOAP para la invocación de un WS, desde el SLEE hacia la red subyacente, y de terminación de sesión entre el UA IMS y el WS, respectivamente.

En el estado `FINAL` la máquina termina su ejecución y retorna al estado `INITIAL`.

2. La FSM Tool utiliza la especificación de la máquina de estados para generar automáticamente una clase Java del SBB de la FSM. Para generar correctamente el código se necesitó:
   - Fijar las propiedades para la plantilla de generación de código FSM SBB de la FSM Tool (archivo *build.properties*, y algunas otras propiedades en el *build.xml*).
   - Configurar el archivo *build.xml* de Ant, para utilizar la FSM Tool Ant task:
     - Agregar al archivo *build.xml* una tarea `<vfsmgen>`, y fijar las propiedades de la tarea para el componente que se está desarrollando.
     - Ejecutar Ant en el objetivo o target que incluye la tarea `<vfsmgen>`.
     - Analizar los errores que reporta la FSM Tool cuando se hace el parseo de la especificación, corregirlos, y luego volver a correr la target Ant.

   En este proceso se generaron las clases: *SipMiddStateMachineSbb.java*, *SipUASMiddStateMachineSbb.java*, *SoapMiddStateMachineSbb.java*, y *SoapUASMiddStateMachineSbb.java*. Se recomienda la utilización de la herramienta de OpenCloud XDoclet para generar los descriptores de despliegue para los SBBs, ya que la clase SBB generada declara los campos CMP utilizando las etiquetas de XDoclet. Si no se utiliza XDoclet se necesitará gestionar a mano los descriptores de despliegue del SBB extendido. También se tendría que incluir los campos CMP `StoredState` y `EncodedVirtualInputRegister` en el descriptor de despliegue del SBB.

3. La especificación VFSM define el comportamiento abstracto del componente. Para hacer que funcione, se debió extender la clase Java FSM SBB generada, en el paso 2, con una clase SBB que implementará las acciones definidas en la especificación de la FSM. Esta nueva clase SBB debe: mapear los eventos recibidos a las acciones de entrada, y ejecutar la máquina de estados; e implementar los métodos de acción que la clase generada declara abstractamente. Los desarrolladores de JAIN SLEE están acostumbrados a escribir la lógica en los métodos manejadores de eventos del SBB. La FSM Tool cambia este estilo, e implementa todo el código de la lógica del servicio en los métodos de las acciones. Con la FSM Tool sólo se utilizan los métodos manejadores de eventos para el mapeo de los eventos a las entradas de la máquina de estados, llamando al método `receivedInput(Input, Object, ActivityContextInterface)`. Es importante tener en cuenta que este método no puede ser llamado en un método de acción, sino se ocasionaría una excepción en tiempo de ejecución. Por lo tanto:
   - El SBB recibe un evento, a través del RA correspondiente, en el método manejador de evento respectivo,
   - Por medio de la llamada al método `receivedInput` de la clase Java del SBB de la FSM se informa a la VFSM de su llegada,
   - la VFSM redirige el procesamiento de dicho evento al método de acción especificado.

4. Finalmente, se compiló, empaquetó y desplegó a MIDDIS utilizando los métodos estándar para el despliegue completo de los SBB. El ejecutable del software creado consiste en los archivos fuente, las librerías, la unidad desplegable, y el descriptor del despliegue del servicio.

- **Servicio de Registro de un WS en IMS, y Servicio de Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI**

Para dar soporte a los procesos de Registro de un WS en IMS, y a la Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI, la definición del servicio y de la jerarquía de SBBs que implementan estas funcionalidades del SSMIDD y del SSREGS de MIDDIS, especificada en los descriptores de despliegue a través XDoclet, son:

- **Definición del Servicio: MiddisSOAPService**

```
@slee.service
 description="MiddisSOAPService"
 name="MiddisSOAPService"
 vendor="co.edu.unicauca.git"
 version="@VERSIONSTR@"
 default-priority="0"
```

En la clase del SBB se utiliza la etiqueta `@slee.service` para generar el descriptor de despliegue del servicio y para identificarla como una clase abstracta de un SBB raíz. A continuación se describen sus parámetros:

- `description`: especifica la descripción del servicio.
- `name`: especifica el nombre del servicio. Este valor se asigna al elemento `<service-name>`.
- `vendor`: especifica el vendedor del servicio. Este valor se asigna al elemento `<service-vendor>`.
- `version`: especifica la versión del servicio. Este valor se asigna al elemento `<service-version>`.
- `default-priority`: entero que especifica, para las relaciones padre a hijo, la prioridad por defecto en la entrega de eventos desde el SLEE hacia el SBB raíz.

- **Definición de la Jerarquía de SBB:** en el SBB raíz `SoapMiddSbb`

Este SBB maneja la lógica de la señalización SOAP. Por un lado, recibe eventos de petición del SOAP RA, provenientes del componente SOA/WS, y genera órdenes de mediación SOAP/SIP que dirige hacia el `MiddSbb`, y por otro lado recibe los resultados de la mediación SOAP/SIP los cuales son convertidos en eventos de respuesta de tipo SOAP que se envían hacia el componente SOA/WS, a través del SOAP RA.

- *SBB Raíz*:

```
@slee.sbb
 id="soapmiddsbb"
 name="SoapMiddSbb"
 vendor="co.edu.unicauca.git"
```

- *SBB Hijo*:

```
@slee.sbb-ref
 sbb-id="middsbb"
 sbb-name="Middleware Sbb"
 sbb-vendor="co.edu.unicauca.git"
```

```
version="0.1"
reentrant=True
```

```
sbb-version="0.1"
sbb-alias="MIDDSBB"
```

En la clase del SBB raíz se utiliza la etiqueta `@slee.sbb` para identificarla como una clase abstracta de SBB. A continuación se describen sus parámetros:

- `id`: especifica el identificador del SBB. Este valor se asigna al atributo `id` del elemento `<sbb>`.
- `name`: especifica el nombre del SBB. Este valor se asigna al elemento `<sbb-name>`.
- `vendor`: especifica el vendedor del SBB. Este valor se asigna al elemento `<sbb-vendor>`.
- `version`: especifica la versión del SBB. Este valor se asigna al elemento `<sbb-version>`.

El parámetro fundamental que permite establecer en el descriptor de despliegue, de los SBB, los bucles de retorno desde un SBB hijo hacia un SBB padre, dentro del mismo contexto de la transacción, es:

`reentrant=True`

Un ejemplo de un bucle de retorno es cuando una entidad de SBB A recibe una invocación a un método, luego A llama a la entidad del SBB B, y finalmente el SBB B llama de regreso al SBB A, dentro del mismo contexto de transacción. El método invocado por el bucle de retorno comparte el contexto actual de ejecución (lo cual incluye a la transacción), junto con la invocación inicial de la entidad del SBB A (sección 6.11 de (Sun Microsystems y OpenCloud, 2009)).

En la clase de un SBB se pueden especificar SBB(s) padre(s) o hijo(s), de la misma, a través de la etiqueta `@slee.sbb-ref`, la cual genera los elementos `<sbb-ref>` en el descriptor de despliegue. En este caso, al utilizar dicha etiqueta en la clase del SBB raíz, se está especificando un SBB hijo. A continuación se describen sus parámetros:

- `sbb-id`: especifica el identificador de un SBB, en la misma ruta de generación del doclet.
- `sbb-name`: especifica el nombre del SBB referenciado.
- `sbb-vendor`: especifica el vendedor del SBB referenciado.
- `sbb-version`: especifica la versión del SBB referenciado.
- `sbb-alias`: especifica el alias para otras referencias en el descriptor. Este valor se asigna al elemento `<sbb-alias-ref>`.

o **Definición de la Jerarquía de SBB:** en el SBB `MiddSbb`

Este SBB realiza la mediación a nivel de señalización y control entre SIP y SOAP, adaptando todas las peticiones y respuestas de un protocolo de comunicación (por ejemplo SIP) al otro protocolo de comunicación (por ejemplo SOAP). Para el Servicio de Registro de un WS en IMS, ver Figura B20, y el Servicio de Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI, ver Figura B21, por un lado recibe órdenes de mediación SIP/SOAP desde el `SoapMiddSbb`, las procesa y ejecuta utilizando su propia

funcionalidad y/o la del `SipMiddSbb`, y por otro lado recibe los resultados de dicha mediación, que pueden provenir del `SipMiddSbb`, los cuales son convertidos en órdenes de respuesta que envía de regreso al `SoapMiddSbb`. Este SBB se comporta como un Agente de Usuario Extremo a Extremo (Back-to-Back User Agent, B2BUA), es decir, como una entidad lógica que recibe una solicitud, la procesa como si fuese un Agente de Usuario Servidor (UAS) y, para determinar cómo contestar la solicitud, actúa como un Agente de Usuario Cliente (UAC) generando la solicitud correspondiente. En este sentido, el SBB debe seguir el estado de la llamada, para lo cual debe dialogar, mediante solicitudes y respuestas, con los UA que intervienen en la llamada.

- *SBB*:

```
@slee.sbb
 id="middsbb"
 name="Middleware Sbb"
 vendor="co.edu.unicauca.git"
 version="0.1"
 reentrant=True
```

- *SBB Padre*:

```
@slee.sbb-ref
 sbb-id="soapmiddsbb"
 sbb-name="SoapMiddSbb"
 sbb-vendor="co.edu.unicauca.git"
 sbb-version="0.1"
 sbb-alias="SOAPMIDDSBB"
```

- *SBB Hijo*:

```
@slee.sbb-ref
 sbb-id="sipmiddsbb"
 sbb-alias="SIPMIDDSBB"
```

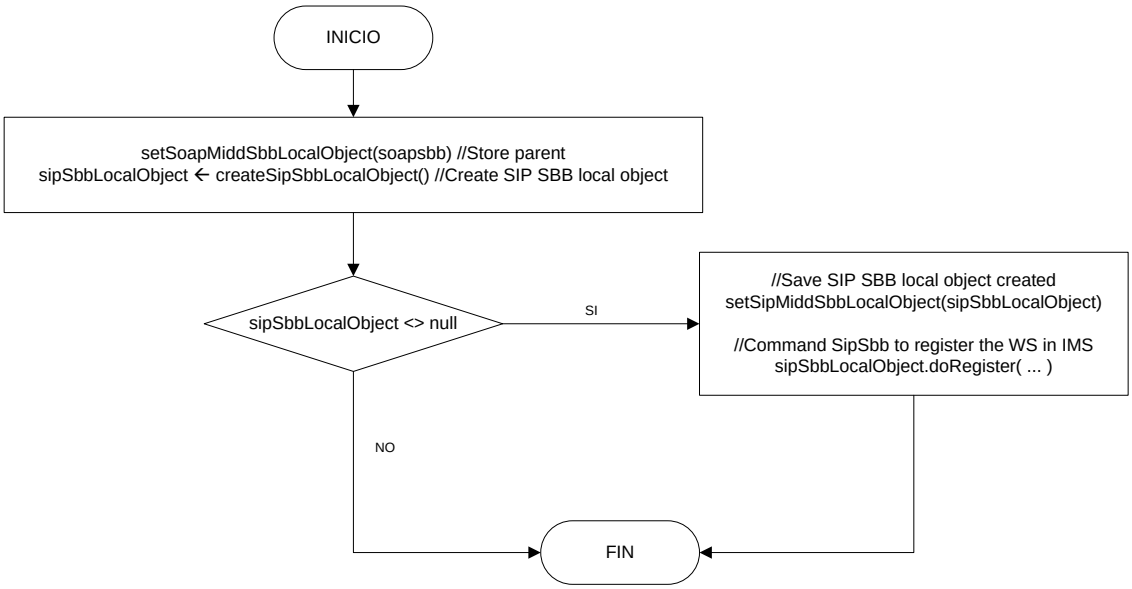


Figura B20. MiddSbb: Diagrama de Flujo para el procesamiento de una petición de registro de un WS en IMS

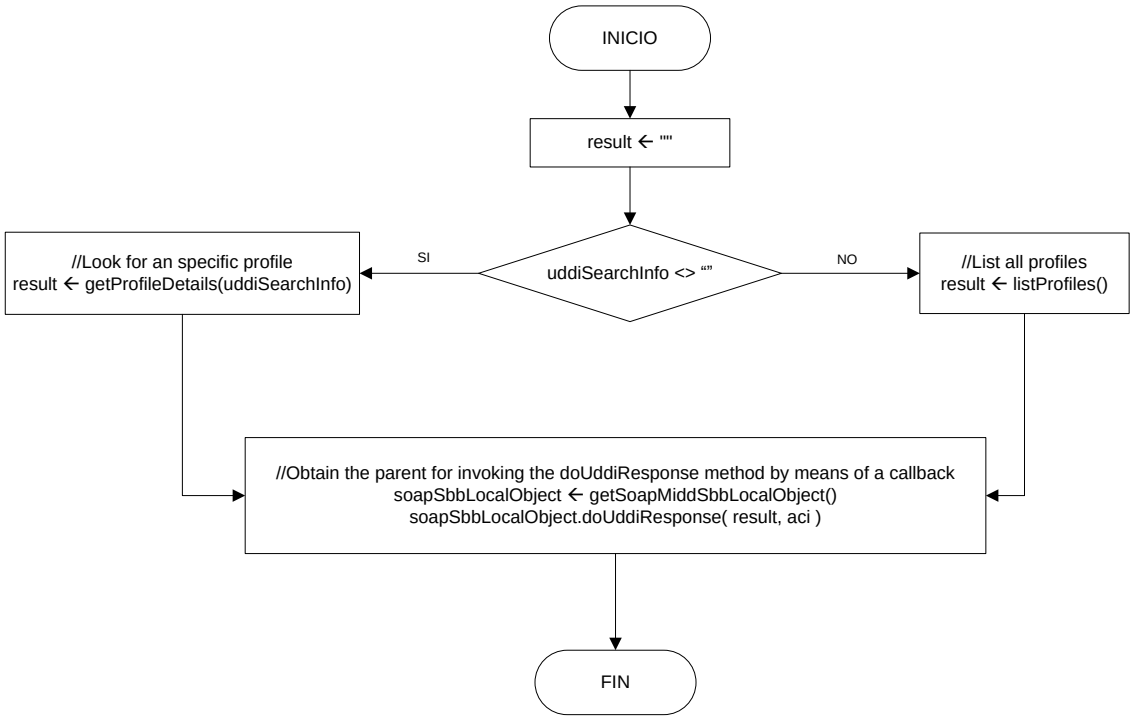


Figura B21. MiddSbb: Diagrama de Flujo para la consulta de la descripción de un Servicio en el registro UDDI de MIDDIS

- **Definición de la Jerarquía de SBB:** en el SBB `SipMiddSbb`

Este SBB maneja la lógica de la señalización SIP. Por un lado, recibe la orden de mediación SOAP/SIP, proveniente del `MiddSbb`, para el registro del WS en IMS, y la procesa y ejecuta (como se muestra en la Figura B22) enviando peticiones de tipo SIP hacia el Núcleo IMS de la red subyacente, a través del SIP RA. Por otro lado, al recibir las respuestas de tipo SIP a esas peticiones las procesa y retorna el resultado al `MiddSbb`, el cual se encarga posteriormente del procesamiento correspondiente.

- *SBB*:

```
@slee.sbb
 id="sipmiddsbb"
 name="SipMiddSbb"
 vendor="co.edu.unicauca.git"
 version="0.1"
```

- *SBB Padre*:

```
@slee.sbb-ref
 sbb-id="middsbb"
 sbb-name="Middleware Sbb"
 sbb-vendor="co.edu.unicauca.git"
 sbb-version="0.1"
 sbb-alias="MIDDSBB"
```

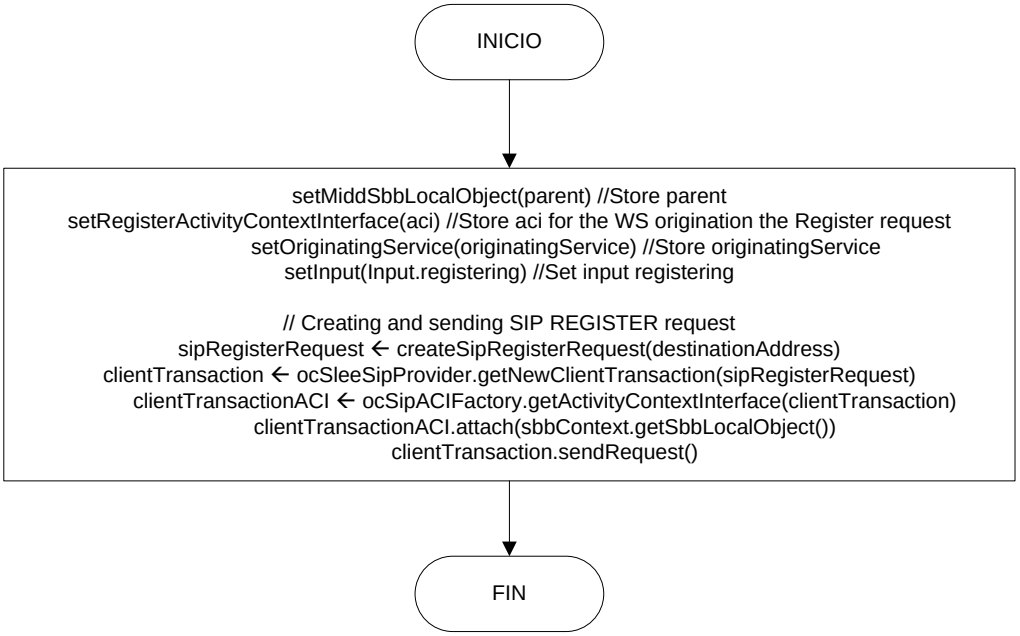


Figura B22. SipMiddSbb: Diagrama de Flujo para el registro del WS en IMS

En la Figura B23 se muestra el árbol de entidades de SBB para los servicios de Registro de un WS en IMS, y de Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI. La comunicación entre los SBB se puede realizar a través de la Interfaz Local de cada SBB o mediante eventos personalizados. En este caso, para componer los servicios de los SSMIDD y SSREGS la comunicación entre SBB se implementó través de la Interfaz del Objeto Local de cada SBB, ya que hacen parte del mismo árbol de entidades de SBB.

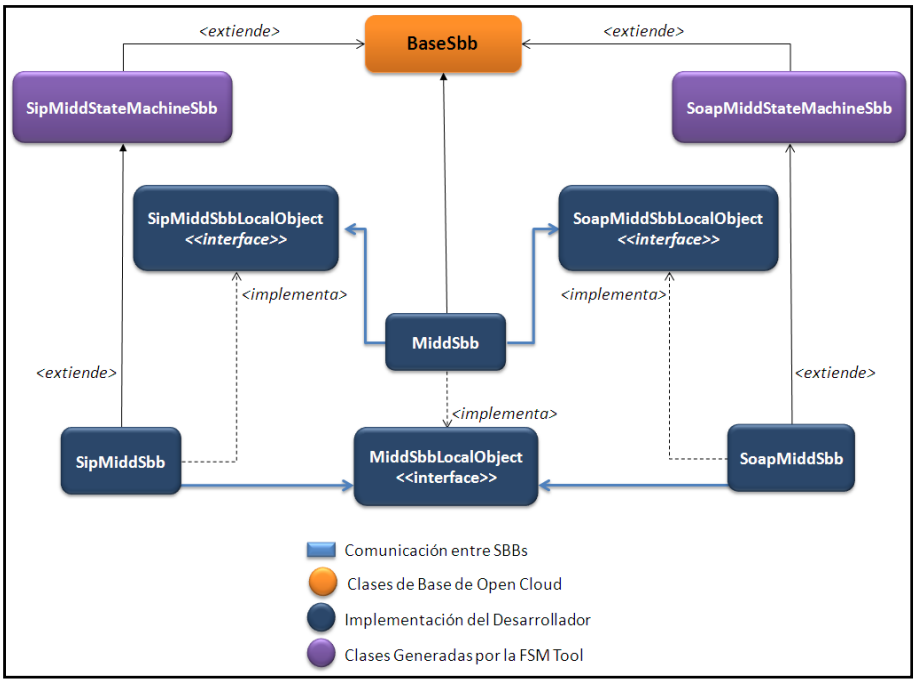


Figura B23. Árbol de entidades de SBB para los servicios de Registro de un WS en IMS, y de Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI

Una entidad de SBB puede invocar otra entidad de SBB de manera síncrona a través de la Interfaz Local del SBB de destino, Capítulo 5 de (Sun Microsystems y OpenCloud, 2009). Esta

interfaz es conocida como la interfaz local del SBB, debido a que el objeto del SBB que representa al llamante y el objeto del SBB que representa al llamado deben estar localizados en la misma Máquina Virtual de Java. Por lo tanto, con el fin de que un SBB sea invocado de manera síncrona, éste debe declarar tanto la interfaz local del SBB como los métodos que pueden ser invocados síncronamente. Así mismo, una característica fundamental de la invocación síncrona a métodos a través de las Interfaces Locales de los SBB es que éstas no permiten que SBB de diferentes servicios se comuniquen entre sí.

En un escenario típico, el objeto de un SBB, que representa la entidad de un SBB padre, crea una entidad de un SBB hijo por medio de la invocación del método `create` en uno de sus objetos `ChildRelation`. Éste método retorna un objeto que implementa la interfaz local de la entidad de SBB hija; es decir, un objeto local que representa la entidad de SBB hija recién creada. Tener en cuenta que cero o más objetos locales de un SBB pueden representar la misma entidad de SBB.

Cuando el SBB padre, u otro SBB, invoca en el objeto local de un SBB un método definido por el desarrollador, dicho objeto local delega la invocación a un objeto SBB local que represente la entidad SBB de destino. Los métodos de la Interfaz Local de un SBB se ejecutan en el contexto de la transacción de los métodos llamantes, por lo tanto ellos no involucran una nueva transacción. El SLEE es el responsable del suministro de un objeto SBB local adecuado para recibir esta invocación. Si no existe, el SLEE crea un nuevo objeto del SBB, u obtiene uno existente que se encuentre en el estado Pooled, y luego lo asigna a la entidad del SBB de destino.

- **Implementación de la Gestión de las descripciones de los Servicios IMS y Web basada en el modelo UDDI**

Los procesos de publicación, consulta y modificación de las descripciones de los Servicios IMS y Web en MIDDIS se basan en el modelo UDDI de los WS (Oasis, 2002), y constituyen uno de los principales aportes del presente trabajo de grado de maestría.

Los datos necesarios para el registro de la descripción de un servicio en el UDDI de MIDDIS son:

```
<uddiInfo>
  <businessEntityName>Entity Name</businessEntityName>
  <businessEntityDescription>Entity Description
                                </businessEntityDescription>
  <businessEntityContactPhone>Entity Contact Phone
                                </businessEntityContactPhone>
  <businessEntityContactEmail>Entity Contact Email
                                </businessEntityContactEmail>
  <businessServiceName>Service Name</businessServiceName>
  <businessServiceDescription>Service Description
                                </businessServiceDescription>
  <bindingTemplateLocation>Template Location</bindingTemplateLocation>
  <bindingTemplateArchitecture>Template Architecture
                                </bindingTemplateArchitecture>
  <bindingTemplateTModel>Template TModel</bindingTemplateTModel>
</uddiInfo>
```

- **businessEntityName**: Compañía que desarrolla el servicio.
- **businessEntityDescription**: Descripción de la compañía que desarrolla el servicio.
- **businessEntityContactPhone**: Información de contacto de la compañía.
- **businessEntityContactEmail**: Información de contacto de la compañía.
- **businessServiceName**: Información técnica del servicio.
- **businessServiceDescription**: Información que describe técnicamente el servicio.
- **bindingTemplateLocation**: Hace parte de la plantilla que almacena la descripción técnica del servicio concerniente a su localización. Es decir, la dirección o localización donde se puede encontrar la implementación actual del servicio.
- **bindingTemplateArchitecture**: Hace parte de la plantilla que almacena la descripción técnica del servicio concerniente a la descripción de su arquitectura.
- **bindingTemplateTModel**: Hace parte de la plantilla que almacena las especificaciones técnicas que describen la naturaleza de un servicio, para el caso de los WS corresponde a la WSDL.

En MIDDIS, la implementación de la publicación, consulta y modificación de las descripciones de los servicios basada en UDDI se realizó por medio de los Perfiles de JAIN SLEE, como se especifican en la versión 1.1 del API (Capítulo 10 de (Sun Microsystems y OpenCloud, 2009)).

Los Perfiles son objetos que contienen datos requeridos por un componente (por ejemplo un SBB) para realizar sus funciones. Entre dichos datos se pueden encontrar datos de configuración o datos del suscriptor. La Especificación de un Perfil describe el esquema de las Tablas del Perfil. En dicha especificación se definen el conjunto de atributos para cada Tabla del Perfil. Cada Perfil creado en las Tablas del Perfil tienen los mismos atributos. La especificación 1.1 del SLEE define 2 tipos de consultas o *queries*: estática y dinámica. Toda consulta estática que sea requerida se declara en la especificación del perfil; mientras, que las consultas dinámicas pueden ser declaradas después de que se haya desplegado una Especificación de un Perfil.

La Interfaz CMP del Perfil define la colección de atributos persistentes de la Especificación del Perfil. Todos los atributos persistentes que están contenidos en un Perfil deben ser incluidos en la Interfaz CMP del Perfil. La etiqueta de XDoclet requerida para generar el descriptor de despliegue de la Especificación del Perfil para la gestión UDDI de MIDDIS es:

```
@slee.profile-spec
 id="midduddiprofile"
 description="Middleware Profile for UDDI"
 name="MiddUDDIProfile"
 vendor="co.edu.unicauca.git"
 version="0.1"
 profile-read-only="False"
```

Esta etiqueta identifica la clase como una Interfaz CMP del Perfil. El atributo `profile-read-only` permite definir la visualización del perfil; es decir, que a través del valor booleano especificado en este atributo se determina si el Perfil tiene una visualización de solo lectura, o de lectura y escritura. Tener en cuenta que su valor por defecto es “True”, por lo cual

si se necesita crear, eliminar o modificar un perfil, en tiempo de ejecución del servicio, el JSLEE generará las excepciones correspondientes y no permitirá dichas operaciones.

Para poder utilizar la Especificación de un Perfil dentro de los SBB que componen el servicio JSLEE, se debe definir la referencia al Perfil en el descriptor de despliegue de los SBB. XDoclet permite realizar esta referencia a través de la siguiente etiqueta, definida en cualquiera de los SBB que conforman el servicio:

```
@slee.profile-spec-ref
 profile-spec-id="midduddiprofile"
 profile-spec-alias="MIDDUDDIPROFILE"
```

Los métodos de los Perfiles JSLEE permiten realizar las operaciones de creación, eliminación, y consulta de los datos de un perfil. Algunos de ellos se describen a continuación:

```
//Creación de un perfil:
MiddUDDIProfileCMP profile =
(MiddUDDIProfileCMP)getProfileTable().create( profileName );
//Adición de los Atributos del perfil:
profile.setBusinessEName( entityname );
            profile.setBusinessEDescription( entitydesc );
…
//Eliminación de un perfil:
//El objeto TABLE_NAME, de tipo cadena, representa el nombre de la
//tabla de perfiles UDDI dentro de MIDDIS
getProfileFacility().getProfileTable( TABLE_NAME ).remove(
profileName );

//Consulta del dato businessEntityName de un perfil:
MiddUDDIProfileCMP profile =
(MiddUDDIProfileCMP)getProfileTable().find( profileName );
String response = "";
response = "businessEntityName: " + profile.getBusinessEName();
```

Es importante tener en cuenta que para poder utilizar los perfiles dentro del JSLEE debe existir la tabla de perfiles correspondiente. En MIDDIS dicha tabla es creada por medio de la tarea de gestión del SLEE denominada `createprofiletable`.

Para la ejecución de la Gestión, basada en UDDI, de las descripciones de los Servicios IMS y Web se utilizan peticiones de tipo SOAP. Por una parte, el registro o modificación de la descripción de un servicio se ejecuta por medio de un documento XML, que representa el Sobre de la petición SOAP, donde se incluyen la operación, en este caso `setUddi`, el nombre del perfil dentro de la etiqueta `<webServiceName>`, y los parámetros correspondientes a la descripción misma del servicio, de acuerdo al modelo UDDI de MIDDIS. EL nombre del perfil incluido dentro de esta petición SOAP permitirá determinar primero la existencia del mismo dentro del registro UDDI de MIDDIS, y posterior a este análisis permitirá seleccionar la operación a ejecutar, que puede ser la adición de un nuevo perfil o la actualización de uno existente. A continuación, se muestra un ejemplo genérico de este tipo de documento XML:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <soa>
      <operation>setUddi</operation>
      <webServiceName>WSfootwearshopService</webServiceName>
```

```
      <uddiInfo>
            <businessEntityName>Entity Name</businessEntityName>
            <businessEntityDescription>Entity
Description</businessEntityDescription>
            <businessEntityContactPhone>Entity Contact
Phone</businessEntityContactPhone>
            <businessEntityContactEmail>Entity Contact
Email</businessEntityContactEmail>
            <businessServiceName>Service Name</businessServiceName>
            <businessServiceDescription>Service
Description</businessServiceDescription>
            <bindingTemplateLocation>Template
Location</bindingTemplateLocation>
            <bindingTemplateArchitecture>Template
Architecture</bindingTemplateArchitecture>
            <bindingTemplateTModel>Template
TModel</bindingTemplateTModel>
      </uddiInfo>
    </soa>
  </Body>
</Envelope>
```

Por otra parte, el proceso de consulta de la descripción de un servicio se ejecuta también por medio de un documento XML, que representa el Sobre de la petición SOAP, donde se incluyen la operación, en este caso `getUddi`, y una etiqueta `<webServiceName>` que puede contener el nombre del perfil a consultar, o el carácter "−", por medio del cual se consulta la lista de los nombres de los perfiles registrados actualmente en MIDDIS. A continuación, se muestra un ejemplo de este tipo de documento XML:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
 <Body>
      <soa>
            <operation>getUddi</operation>
            <webServiceName>WSfootwearshopService</webServiceName>
      </soa>
 </Body>
</Envelope>
```

- **Servicio de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS**

Para dar soporte a los procesos de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS, la definición del servicio y de la jerarquía de SBB que implementan estas funcionalidades del SSMIDD y del SSREGS de MIDDIS, especificadas en los descriptores de despliegue a través XDoclet, son:

  - **Definición del Servicio: MiddisSIPService**

```
@slee.service
 description="MiddisSIPService"
 name="MiddisSIPService"
 vendor="co.edu.unicauca.git"
 version="@VERSIONSTR@"
 default-priority="0"
```

- **Definición de la Jerarquía de SBB:** en el SBB raíz `SipUASMiddSbb`

Este SBB maneja la lógica de la señalización SIP. Por un lado, recibe eventos de petición/respuesta del SIP RA, provenientes del Núcleo de IMS, y genera órdenes de mediación SIP/SOAP que dirige hacia el `MiddSbb`, y por otro lado recibe los resultados de la mediación SIP/SOAP los cuales son convertidos en eventos de petición/respuesta SIP que se envían hacia el Núcleo de IMS, a través del SIP RA. Este SBB se comporta como un UAC y un UAS de SIP.

- *SBB Raíz*:

```
@slee.sbb
 id=" sipuasmiddsbb"
 name="SipUASMiddSbb"
 vendor="co.edu.unicauca.git"
 version="0.1"
 reentrant=True
```

- *SBB Hijo*:

```
@slee.sbb-ref
 sbb-id="middsbb"
 sbb-name="Middleware Sbb"
 sbb-vendor="co.edu.unicauca.git"
 sbb-version="0.1"
 sbb-alias="MIDDSBB"
```

- **Definición de la Jerarquía de SBBs:** en el SBB `MiddSbb.java`

Este SBB realiza la mediación a nivel de señalización y control entre SIP y SOAP, adaptando todas las peticiones y respuestas de un protocolo de comunicación (por ejemplo SIP) al otro protocolo de comunicación (por ejemplo SOAP). Para el Servicio de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS (Figura B24), para el acceso a un WS desde un Servicio IMS, por un lado recibe órdenes de mediación SIP/SOAP desde el `SipUASMiddSbb`, las procesa y ejecuta utilizando su propia funcionalidad y la del `SoapUASMiddSbb`, y por otro lado recibe los resultados de dicha mediación, que pueden provenir del `SoapUASMiddSbb`, los cuales son convertidos en órdenes de respuesta que envía de regreso al `SipUASMiddSbb`. Este SBB se comporta como un B2BUA.

- *SBB*:

```
@slee.sbb
 id="middsbb"
 name="Middleware Sbb"
 vendor="co.edu.unicauca.git"
 version="0.1"
 reentrant=True
```

- *SBB Padre*:

```
@slee.sbb-ref
 sbb-id="sipuasmiddsbb"
 sbb-name="SipUASMiddSbb"
 sbb-vendor="co.edu.unicauca.git"
 sbb-version="0.1"
 sbb-alias="SIPUASMIDDSBB"
```

- *SBB Hijo*:

```
@slee.sbb-ref
 sbb-id="soapuasmiddsbb"
 sbb-alias="SOAPUASMIDDSBB"
```

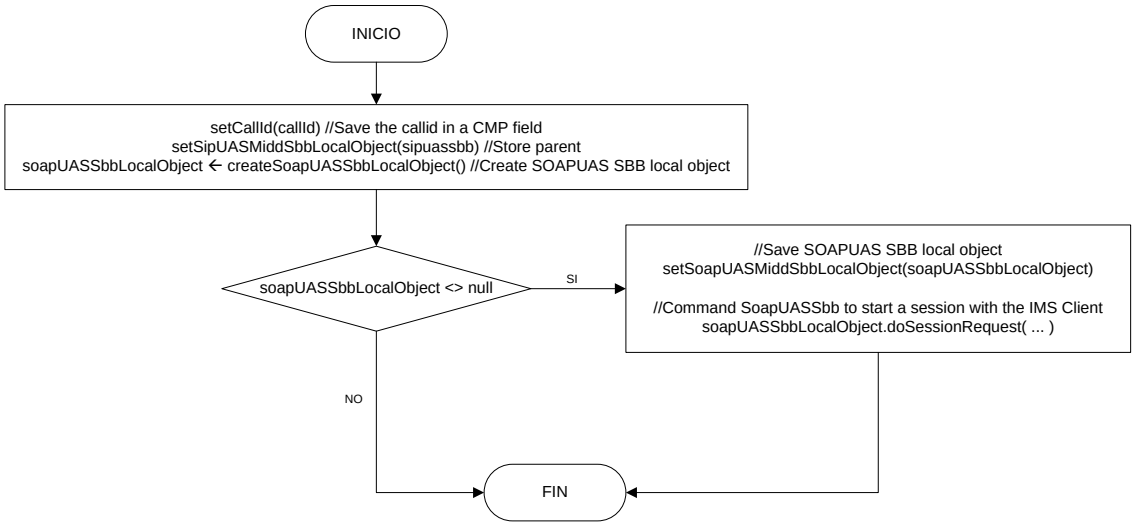


Figura B24. MiddSbb: Diagrama de Flujo para el Inicio de sesión entre un UA IMS y un WS

- **Definición de la Jerarquía de SBB:** en el SBB `SoapUASMiddSbb`

Este SBB maneja la lógica de la señalización SOAP. Por un lado, recibe órdenes de mediación SIP/SOAP provenientes del `MiddSbb` las cuales procesa y ejecuta enviando peticiones de tipo SOAP hacia el componente SOA/WS de la red subyacente, a través del SOAP RA, y por otro lado al recibir las respuestas de tipo SOAP a esas peticiones las procesa y retorna el resultado al `MiddSbb`, el cual se encarga posteriormente del procesamiento correspondiente.

- *SBB*:

```
@slee.sbb
 id="soapuasmiddsbb"
 name="SoapUASMiddSbb"
 vendor="co.edu.unicauca.git"
 version="0.1"
```

- *SBB Padre*:

```
@slee.sbb-ref
 sbb-id="middsbb"
 sbb-name="Middleware Sbb"
 sbb-vendor="co.edu.unicauca.git"
 sbb-version="0.1"
 sbb-alias="MIDDSBB"
```

En la Figura B25 se muestra el árbol de entidades de los SBB para los servicios de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS. La comunicación entre los SBB es a través de la Interfaz del Objeto local de cada SBB, ya que hacen parte del mismo árbol de entidades de SBB.

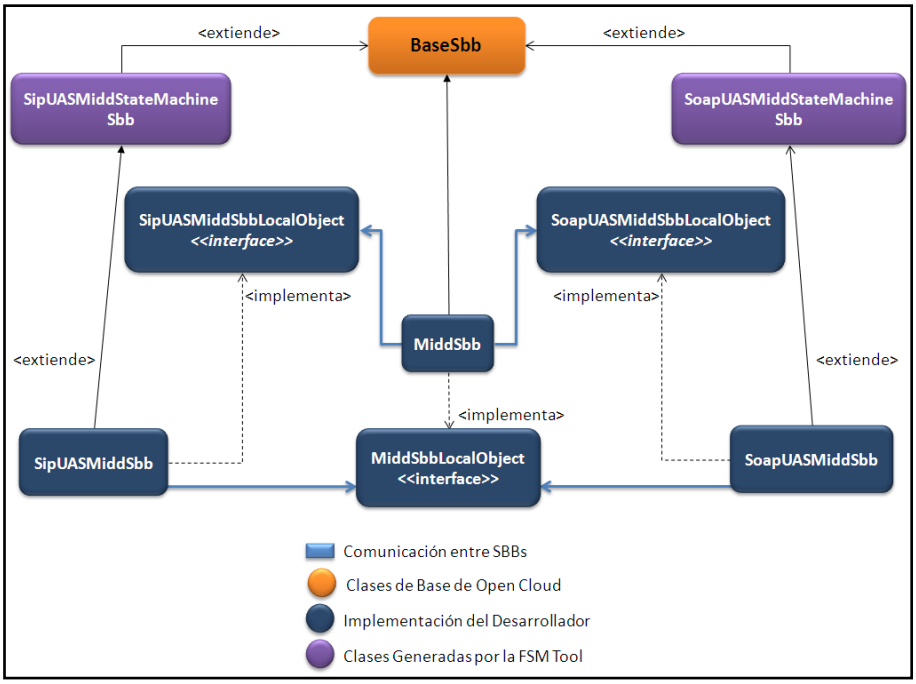


Figura B25. Árbol de entidades de SBB para los servicios de Inicio y mantenimiento de Sesiones entre un Servicio IMS y un WS para el acceso a un WS desde un Servicio IMS

B.6.4. Núcleo de IMS

La Comunidad del Instituto Tecnológico Fraunhofer de Berlin para la investigación y desarrollo de sistemas de comunicaciones móviles, de código abierto, en redes fijas e inalámbricas, FOKUS (Fraunhofer institute Für Offene Kommunications Sisteme), implementó el Open IMS Core, el cual a su vez está inscrito dentro del proyecto "Open IMS Playground @ FOKUS", un marco de desarrollo de aplicaciones para la tecnología IMS, dentro del cual se pueden encontrar diferentes tecnologías de acceso, además de la implementación de referencia de todos los componentes del núcleo de IMS y las herramientas para su gestión. Por tanto, se instaló el Open IMS Core para proporcionar la infraestructura de la red subyacente, correspondiente al Núcleo de IMS, que hace uso de las funcionalidades de la Lógica de Mediación (ubicada en la capa de Aplicaciones de IMS, en el servidor Rhino SLEE). El Open IMS Core proporciona la implementación de los CSCFs, elementos centrales para el enrutamiento de la señalización IMS, y del Servidor Local del Suscriptor (Home Subscriber Server, HSS), para gestionar los perfiles de usuario y las reglas de enrutamiento asociadas. (Fraunhofer FOKUS, 2009)

Entre las consideraciones más importantes que se deben tener en cuenta para la interoperación, a nivel de señalización SIP-IMS, entre el OpenIMSCore y el Rhino SLEE se tienen:

- La configuración de la propiedad *OUTBOUND-PROXY* de SIP en el SBB (`SipMiddSbb`) encargado del proceso de Registro del WS en IMS.
- La creación de las P-Headers, y las cabeceras SIP adicionales, necesarias para los procesos de Autorización y Autenticación de un UA en el OpenIMSCore:

AuthorizationHeader, SupportedHeader, P-Preferred-Identity, P-Access-Network-Info, Privacy, User-Agent, AllowHeader.

B.7. DESPLIEGUE DE MIDDIS

B.7.1. Ejecución de Rhino SDK

1- En el `$RHINO-HOME`:

```
#./init-management-db-sh postgres
```

2- Iniciar Rhino SDK:

```
#./start-rhino.sh
```

Para detenerlo:

```
#./stop-rhino.sh --nice
```

3- Iniciar Consola Cliente de Rhino. En el `$RHINO-HOME`:

```
# cd /client/bin/
# ./rhino-console
```

B.7.2. Despliegue del SIP RA

En el `$RHINO-HOME`:

```
# cd /examples/sip-examples-2.2_06/
# ant deploysipra
```

B.7.3. Despliegue del SOAP RA

En el `$RHINO-HOME`:

```
# cd /examples/soap-2.1/
# ant deploysoapra
```

B.7.3. Despliegue de MIDDIS

1- Copiar el proyecto `middis` en la ruta `$RHINO-HOME/fsmtool/middis/`.

2- Compilar MIDDIS. En el `$RHINO-HOME`:

```
# cd /fsmtool/middis/
# ant build-service
```

3- Desplegar MIDDIS. En el `$RHINO-HOME`:

```
# cd /fsmtool/middis/
# ant deploy-middis-service
```

B.8. EJECUCIÓN DE MIDDIS

B.8.1. Ejecutar el OpenIMSCore

Ejecutar las CSCF:

```
# sudo su
# cd opt/OpenIMSCore
```

En consolas, o pestañas diferentes:

```
# ./pcscf.sh
# ./icscf.sh
# ./scscf.sh
```

Finalmente, ejecutar el FHoSS:

```
# sudo su
# cd /opt/OpenIMSCore/FHoSS/deploy
# ./startup.sh
```

B.8.2. Desplegar WSfootwearshopService

Desplegar el servicio `WSfootwearshopService` en el servidor de aplicaciones Glassfish. De acuerdo al entorno de pruebas configurado, el servidor Glassfish despliega el WS en la ruta:

```
http://192.168.0.1:8080/ConvergedServices/WSfootwearshopService
```

B.8.3. Ejecutar y Registrar el Cliente IMS

1- En el proyecto Java, `IMSWSConvergedClient`, ejecutar el Cliente IMS a través del archivo: `IMSClient.java`, localizado en la ruta: `/src/co/edu/unicauca/git/ims/gui/client/`.

2- Iniciar la pila de SIP en el Cliente IMS haciendo clic en el botón *Start SIP*.

3- Registrar el Cliente IMS en el IMS a través del botón *REGISTER*. El resultado de los anteriores pasos se muestra en la Figura B26.

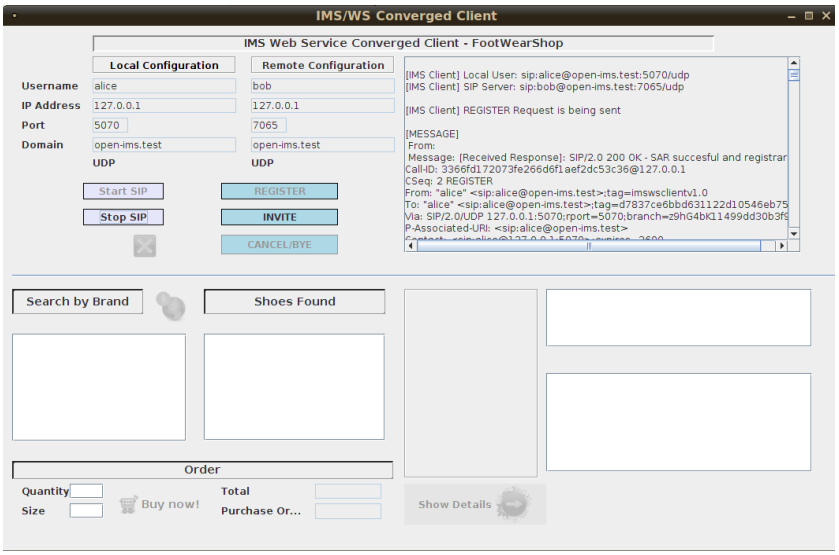


Figura B26. Ejecución y Registro del Cliente IMS

B.8.4. Registrar el WS en IMS a través de MIDDIS

En el $RHINO-HOME:

```
# cd /fsmtool/middis/
# ant run-registertest-client
```

El resultado se muestra en la Figura B27:

```
root@xvelasco:/home/ximena/rhino/RhinoSDK/fsmtool/middis# ant run-registertest-client
Buildfile: /home/ximena/rhino/RhinoSDK/fsmtool/middis/build.xml

compile-test-client:

run-registertest-client:
     [java] <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOA
P-ENV:Header/><SOAP-ENV:Body><soapmidd>Success SOAP Response</soapmidd></SOAP-ENV:Body></SOAP-
ENV:Envelope>

BUILD SUCCESSFUL
```

Figura B27. Registro de un WS en IMS a través de MIDDIS

B.8.5. Iniciar Sesión entre el Cliente IMS y el WS través de MIDDIS

Generar la petición de inicio de sesión desde el Cliente IMS a través del botón INVITE. El resultado exitoso del proceso de inicio de sesión, que se muestra en la Figura B28, habilita el botón de búsqueda de productos por marca. A su vez, es habilitado el botón *CANCEL/BYE* para finalizar la sesión.

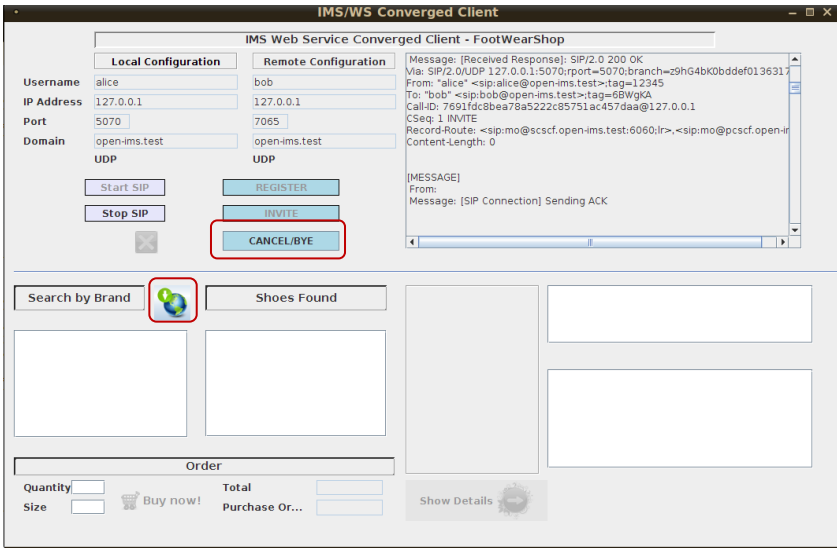


Figura B28. Inicio exitoso de Sesión entre el Cliente IMS y el WS través de MIDDIS

B.8.6. Acceso al WS desde el Cliente IMS a través de MIDDIS

La invocación de cada una de las funcionalidades proporcionadas por el WS se puede ejecutar de las siguientes maneras:

1- Consulta de las marcas de los productos. Se realiza al hacer clic en el botón:

2- Consulta de los productos por marca. Se realiza haciendo doble clic en cualquiera de las marcas disponibles de la lista de *Search by Brand*, Figuras B29 y B30:

Figura B29. Consulta de los productos por marca

Figura B30. Consulta de los productos por marca

3- Consulta de los datos básicos de un producto. Como resultado del paso 2 se genera la lista de los productos disponibles por marca. Los datos básicos de cada producto se obtienen al hacer doble clic en cualquiera de los productos disponibles de la lista *Shoes Found*, Figura B31:

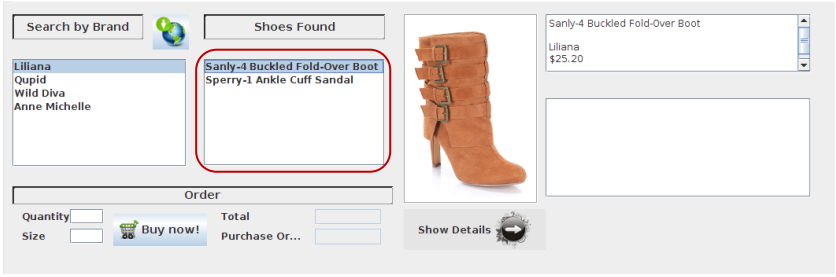


Figura B31. Consulta de los datos básicos de un producto

4- Consulta de los detalles de un producto. Una vez se haya seleccionado un producto, se puede consultar sus detalles, Figura B32, a través del botón *Show Details*:

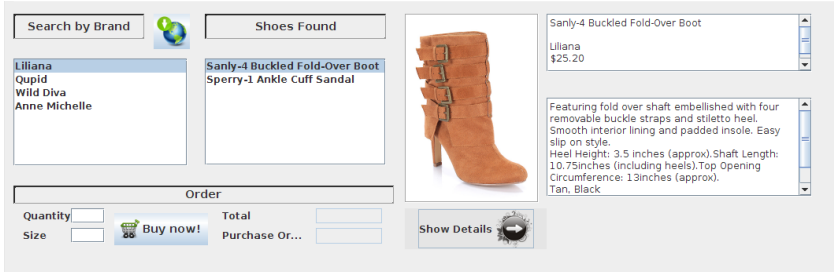


Figura B32. Consulta de los detalles de un producto

5- Generación de una orden de compra. Ingresar los datos necesarios para realizar una compra (*Quantity*, *Size*), Figura B33, y hacer clic en el botón *Buy now!*:

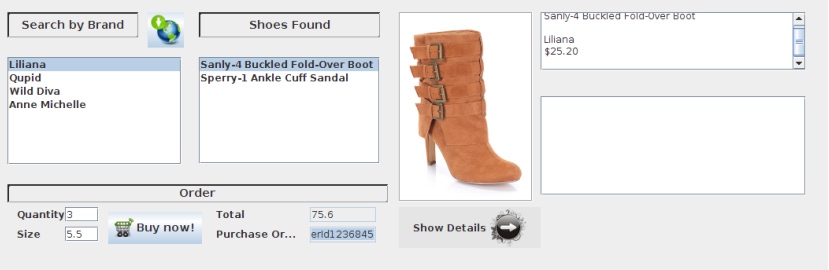


Figura B33. Generación de una orden de compra

Como resultado de la generación de la orden de compra se obtiene el total a pagar (campo *Total*) y el identificador de la compra (campo *Purchase Order ID*).

B.8.7. Registro UDDI

En el $RHINO-HOME:

```
# cd /fsmtool/middis/
# ant run-regudditest-client
```

El resultado se muestra en la Figura B34:

```
root@xvelasco:/home/ximena/rhino/RhinoSDK/fsmtool/middis# ant run-regudditest-client
Buildfile: /home/ximena/rhino/RhinoSDK/fsmtool/middis/build.xml

compile-test-client:

run-regudditest-client:
     [java] <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOA
P-ENV:Header/><SOAP-ENV:Body><soapmidd>Success SOAP Response</soapmidd></SOAP-ENV:Body></SOAP-
ENV:Envelope>

BUILD SUCCESSFUL
```

Figura B34. Registro UDDI

B.8.8. Consulta UDDI

1- Para consultar la lista de servicios registrados en el UDDI de MIDDIS, en el $RHINO-HOME:

```
# cd /fsmtool/middis/
# ant run-searchtesta-client
```

El resultado se muestra en la Figura B35:

```
root@xvelasco:/home/ximena/rhino/RhinoSDK/fsmtool/middis# ant run-searchtesta-client
Buildfile: /home/ximena/rhino/RhinoSDK/fsmtool/middis/build.xml

compile-test-client:

run-searchtesta-client:
     [java] <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOA
P-ENV:Header/><SOAP-ENV:Body><soapmidduddi>Profile List:  - profile name: "WSfootwearshopServi
ce"</soapmidduddi></SOAP-ENV:Body></SOAP-ENV:Envelope>

BUILD SUCCESSFUL
```

Figura B35. Consulta de servicios registrados en el UDDI de MIDDIS

1- Para consultar de la descripción de un servicio registrado en el UDDI de MIDDIS, en el $RHINO-HOME:

```
# cd /fsmtool/middis/
```

```
# ant run-searchtestb-client
```

El resultado se muestra en la Figura B36:

Figura B36. Consulta de la descripción de un servicio registrado en el UDDI de MIDDIS

B.8.9. Finalizar la Sesión entre el Cliente IMS y el WS través de MIDDIS

Hacer clic en el botón *CANCEL/BYE*, y para terminar la aplicación del Cliente IMS, antes de cerrar la ventana, es necesario detener la Pila de SIP mediante el botón *Stop SIP*.

B.9. UTILIZACIÓN DE MIDDIS

En MIDDIS se han implementado los servicios necesarios para el acceso a servicios convergentes IMS/SOA. Dentro del primer grupo se encuentran los servicios para el registro de un WS en IMS, y para el inicio y finalización de la sesión entre un UA IMS y un WS, los cuales son servicios básicos que deben ejecutarse antes de obtener las capacidades para la interacción entre servicios basados en IMS y SOA. Dentro del segundo grupo de servicios se encuentra un piloto de servicio convergente, específico, que permite a un Servicio IMS acceder a las capacidades de uno Web. El tercer grupo de servicios lo conforman aquellos utilizados para la gestión de descripciones de servicios en MIDDIS, basada en UDDI.

Para empezar a utilizar MIDDIS, el desarrollador debe reutilizar el primer grupo de servicios, es decir los servicios básicos de señalización SIP y SOAP que implementa MIDDIS en los SBB correspondientes a cada protocolo, y en el SBB de mediación. Luego, el desarrollador debe implementar el análisis y la composición de mensajes tanto SIP (de tipo `MESSAGE`) como SOAP, para el acceso desde el servicio IMS al WS particular. Esto lo efectúa, por una parte reutilizando las funcionalidades del **`SipUASMiddSbb`** y el **`MiddSbb`**, y, por otra, extendiendo la funcionalidad que se encuentra en el **`SoapUASMiddSbb`**, y adaptándola al acceso del WS particular. Específicamente, el **`SipUASMiddSbb`** recibe el mensaje SIP de tipo `MESSAGE`, y obtiene su contenido, el cual debe constar de la operación de WS a la que se desea acceder y los parámetros necesarios para ejecutar dicha operación. Dicho contenido es enviado automáticamente al **`MiddSbb`**, que a su vez invoca en el **`SoapUASMiddSbb`** el método adecuado de generación de peticiones SOAP de acuerdo al contenido fijado. Una vez el **`SoapUASMiddSbb`** recibe el contenido de la petición SIP original, la analiza y genera, de acuerdo a la operación y parámetros incluidos en ella, la petición SOAP correspondiente, la cual finalmente es enviada al componente SOA/WS de la red subyacente. Esto significa, que todo el proceso de generación de las peticiones SOAP a operaciones de WS particulares, a partir de los datos contenidos en la petición SIP `MESSAGE`, debe implementarse en el **`SoapUASMiddSbb`**.

BIBLIOGRAFIA

Basicevic, L., Popovic, M., Velikic, I. 2010. Use of Finite State Machine Based Framework in Implementation of Communication Protocols – A Case Study. IEEE Computer Society, 2010 Sixth Advanced International Conference on Telecommunications (AICT), 2010, pp. 161-166. 2010.

Fraunhofer FOKUS. 2009. Open IMS Core. [En línea] Fraunhofer FOKUS, 2009. [Citado en: Enero de 2009.] http://www.openimscore.org/.

Oasis. 2002. UDDI Version 2.03 Data Structure Reference. [En línea] Oasis, 19 de Julio de 2002. [Citado en: Agosto de 2010.] http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm#_Toc25130802.

OpenCloud. 2009. FSM Tool Developer's Guide. [En línea] OpenCloud, 5 de Mayo del 2009. [Citado el: 6 de Mayo de 2009.]

OpenCloud. 2010a. Rhino 2.1-03 release. [En línea] OpenCloud, 2010. [Citado en: Abril de 2010.] https://developer.opencloud.com/devportal/display/OCDEV/Rhino+2.1.

OpenCloud. 2010b. SIP Connectivity Pack. [En línea] OpenCloud, 2010. [Citado en: Abril de 2010.] https://developer.opencloud.com/devportal/download/attachments/10552024/sip-connectivity-2.2_06.zip?version=1.

OpenCloud. 2010c. Web Connectivity Pack. [En línea] OpenCloud, 21 de Enero de 2010. [Citado en: Enero de 2010.] https://developer.opencloud.com/devportal/download/attachments/13926487/web-connectivity-2.1.zip?version=1.

Ruiz Rojas, J. 2009. Introducción a JAIN SLEE. [En línea] 2009. [Citado el: 6 de Agosto de 2009.] http://www.slideshare.net/jruizro/developmentinjainsleemay2009?nocache=7891.

Sun Microsystems y OpenCloud. 2009. JAIN SLEE (JSLEE) 1.1 Specification, Final Release. [En línea] 2009. [Citado el: 2 de Mayo de 2009.] http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_JCP-Site/en_US/-/USD/VerifyItem-Start/jslee-1_1-fr-spec.pdf?BundledLineItemUUID=1VlIBe.lNGIAAAEh9Ds1wgvt&OrderID=EO1IBe.l0T0AAAEh4js1wgvt&ProductID=qsJIBe.pUsUAAAEbNVMkexRH&FileName=/jslee-1_1-fr-spec.pdf.