LINEAMIENTOS PARA COMPOSICIÓN DE SERVICIOS DE TELECOMUNICACIONES EN UN ENTORNO JAIN SLEE BASADOS EN SOFTWARE DE LIBRE DISTRIBUCIÓN

Julián Andrés Rojas Meléndez
Jesús David Ramírez Medina

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director:
Dr. Ing. Juan Carlos Corrales

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán, Agosto de 2010

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo I

INTRODUCCIÓN

1.1 Contexto

En los últimos años la Web ha pasado de ofrecer una gran cantidad de información, a proporcionar una variada oferta de recursos y servicios (Web 2.0) entre los que se encuentran comunidades virtuales, redes sociales, blogs, wikis y mashups. La industria de las telecomunicaciones no ha estado ajena a dicha evolución, generando un nuevo modelo conocido como Telco 2.0 [1], el cual relaciona los conceptos, servicios y tecnologías de la Web 2.0 con los servicios de telecomunicaciones tradicionalmente ofrecidos. Bajo este enfoque, las empresas desarrolladoras de servicios de contenido, pueden adicionar capacidades de ubicuidad a sus servicios a través de las funcionalidades de telecomunicaciones o enriquecer los servicios tradicionales ofrecidos por los operadores de telecomunicaciones con capacidades Web 2.0, generando así nuevos servicios de valor agregado.

El desarrollo de este nuevo modelo y de las tecnologías que lo soportan han conducido a una nueva configuración del los procesos de negocio y a aumentar la movilidad y rapidez con que se realizan, modificando las condiciones para la inserción en el mercado de las empresas del sector. Bajo estas nuevas condiciones, se genera la necesidad de que tanto los operadores de telecomunicaciones como las empresas desarrolladoras de servicios de contenido desarrollen capacidades para crear y desplegar nuevos servicios de valor agregado con un bajo *Time-to-Market,* es decir, con tiempos muy cortos para su desarrollo. La mejora del *Time-to-Market* y la capacidad para lanzar al mercado nuevos servicios pasa a ser más importante que el propio ahorro de costos. En la mayoría de los casos, adelantarse a la competencia en el lanzamiento de un servicio genera altos beneficios que amortizan rápidamente los gastos de desarrollo. Sin embargo, el problema actual en la mayoría de las organizaciones empresariales es la incapacidad que presentan para resolver la demanda de nuevos servicios que el negocio solicita, ni siquiera incrementando en gran medida sus costos de desarrollo [2]. Desde este punto de vista, es de gran importancia para los operadores de telecomunicaciones como para las empresas del sector apropiarse de las nuevas tecnologías que permitan la generación de nuevos y mejores servicios TIC de valor agregado.

Dentro de estas nuevas tecnologías se encuentra la especificación JAIN SLEE (Java APIs for Integrated Networks Service Logic Execution Environment) [3,4] la cual presenta una plataforma altamente robusta para la ejecución de servicios convergentes. Debido a su modelo orientado a componentes y en conjunto con los principios que provee la Arquitectura Orientada a Servicios (SOA), es posible llevar a cabo el desarrollo y la composición de servicios a partir de la reutilización de servicios previamente desarrollados. Esto reduce en gran medida el tiempo de lanzamiento al mercado de nuevos y más complejos servicios de valor agregado.

1.2 Escenario de Motivación

Las tecnologías de composición de servicios son un factor estratégico para el sector de las telecomunicaciones ya que permiten hacer uso de los servicios previamente desarrollados como bloques constructores, los cuales trabajando de forma conjunta crean nuevos y más complejos servicios de una forma rápida y flexible, adaptándose a la cambiante dinámica del mercado [5]. Bajo este enfoque, se tiene en cuenta a la especificación JAIN SLEE, la cual define un modelo orientado a componentes para la estructuración de la lógica de aplicaciones y servicios de comunicaciones como un conjunto de componentes reutilizables y orientados a objetos. Esto facilita la composición de dichos componentes en servicios de valor agregado más complejos y sofisticados.

Por otra parte, hoy en día muchas de las aplicaciones de telecomunicaciones tienden a ser creadas como lógica de programas en lenguajes de programación puros como Java, sin embargo, las grandes ventajas propias de las arquitecturas como J2EE (Java 2 Enterprise Edition), que Java ofrece en el mundo de las tecnologías de la información (portabilidad entre plataformas, menores ciclos de implementación por reducción de tiempo en desarrollo y pruebas, etc.) no son suficientes para los servicios de telecomunicaciones, los cuales presentan requerimientos muy diferentes (asíncronos, entorno de ejecución orientado a eventos, tolerancia a fallos, alto rendimiento, etc.). Debido a este tipo de limitaciones, la utilización de Java como herramienta fundamental para el desarrollo de servicios por parte de los operadores de telecomunicaciones y empresas del sector, se encuentra muy limitada [6]. Sin embargo, gracias a iniciativas como JAIN SLEE la cual es el estándar Java para entornos de ejecución de lógica de servicios, es posible para las empresas del sector de las telecomunicaciones, disponer de un entorno de ejecución de servicios de muy baja latencia, alta disponibilidad y orientación a eventos, además de disponer de todas las funcionalidades y ventajas que Java ofrece.

Bajo este punto de vista, la utilización de tecnologías como JAIN SLEE y a través de la composición de servicios, se obtienen grandes ventajas para el desarrollo de nuevos servicios de valor agregado con bajos niveles de *Time-to-Market* generando grandes beneficios para los operadores de telecomunicaciones y las PYMES del sector.

1.3 Definición del Problema

Como se mencionó anteriormente la especificación JAIN SLEE se presenta como una tecnología robusta para el desarrollo, composición y ejecución de servicios que cumplen con los requerimientos tanto funcionales como no funcionales de los servicios de telecomunicaciones. Además, al ser una tecnología fundamentada en el lenguaje de programación Java, es posible incorporar las ventajas y funcionalidades del mundo de las tecnologías de la información provistas por Java, permitiendo el enriquecimiento de los servicios tradicionales de telecomunicaciones. Otra característica previamente mencionada, hace referencia a su modelo orientado a componentes lo que facilita la composición de servicios generando grandes beneficios a las empresas del sector de las telecomunicaciones. Sin embargo, a pesar de que hoy en día las capacidades y beneficios de JAIN SLEE son ampliamente conocidas en el sector de las telecomunicaciones, se plantea una gran limitante a las PYMES del sector debido a que para este tipo de empresas resulta muy difícil tener la capacidad económica que les permita adquirir sistemas propietarios basados en la especificación JAIN SLEE, cosa que las grandes empresas de las telecomunicaciones pueden hacer. De la misma manera, las

herramientas y entornos que permiten el desarrollo y composición de servicios para entornos JAIN SLEE sufren esta misma limitante.

Actualmente, se desarrollan proyectos de libre distribución que implementan la especificación JAIN SLEE como la iniciativa Mobicents [7], lo cual permite a las PYMES del sector tener acceso a las capacidades y beneficios que ofrece JAIN SLEE. De esta misma forma, hoy en día existen algunos entornos para el desarrollo y composición de servicios JAIN SLEE de libre distribución. Sin embargo, el carácter libre de estas herramientas causa que exista muy poca información y/o documentación sobre ellas, limitando la explotación de las capacidades de JAIN SLEE a través de la composición de servicios.

Otro aspecto a tener en cuenta, es que gracias al modelo orientado a componentes de la especificación JAIN SLEE, se facilita la reutilización de componentes y por lo tanto es posible realizar composición de servicios, sin embargo, actualmente no existen unas directrices o lineamientos generales que describan cual es la mejor manera de llevar a cabo la composición de servicios de telecomunicaciones en entornos JAIN SLEE, generando incrementos en los tiempos de desarrollo y despliegue de nuevos servicios de valor agregado.

Teniendo en cuenta las consideraciones anteriores, el presente trabajo de grado pretende dar respuesta a la siguiente pregunta de investigación:

¿Cuáles son los lineamientos a seguir que permitan llevar a cabo la composición de servicios de telecomunicaciones en un entorno JAIN SLEE, basado en herramientas de libre distribución?

1.4 Solución Propuesta

Teniendo en cuenta el contexto y el escenario de motivación previamente presentados, este trabajo de grado pretende proponer una serie de pasos o lineamientos que describan la mejor forma de realizar composición de servicios de telecomunicaciones en entornos JAIN SLEE basándose en herramientas de libre distribución. Para esto se realizará una identificación, descripción y finalmente una evaluación de las diferentes herramientas de libre distribución disponibles actualmente, que permiten la composición de servicios JAIN SLEE.

Por otro lado, para la definición de los lineamientos se tendrá como marco de referencia los principios y conceptos planteados por el modelo para la creación de servicios propuesto por SOA.

1.5 Contribuciones

En el presente trabajo de grado se pueden apreciar las siguientes contribuciones:

- **Base conceptual para el desarrollo de servicios JAIN SLEE:** El presente trabajo presenta una base conceptual relacionada con el uso de la especificación JAIN SLEE y sus diferentes capacidades para el desarrollo de servicios de telecomunicaciones.
- **Análisis, descripción y evaluación de entornos de desarrollo:** Se realiza un análisis y descripción de cada uno de los entornos de desarrollo y composición de

servicios JAIN SLEE de libre distribución que actualmente se encuentran disponibles. Finalmente se evalúan sus capacidades y características funcionales con el fin de determinar cuál es la mejor herramienta para componer servicios JAIN SLEE.

- **Lineamientos para composición de servicios JAIN SLEE:** Este trabajo brinda una serie de pasos o directrices a seguir para realizar composición de servicios de telecomunicaciones en entornos JAIN SLEE, usando como marco de referencia los principios y conceptos de SOA.

- **Integración de tecnologías, especificaciones y estándares:** El presente trabajo agrupa diferentes tecnologías y especificaciones basadas en estándares, ampliamente utilizadas actualmente en el mundo de las telecomunicaciones y de la información, tales como: JAIN SLEE, SIP [8], HTTP [9], MGCP [10], JDBC [11], JMX [12], entre otros.

- **Contribución a la comunidad de software libre:** Proporciona un aporte a la comunidad de software libre debido a la definición de lineamientos para composición de servicios JAIN SLEE, enteramente basado en herramientas y entornos de ejecución de libre distribución.

- **Alternativa para las PYMES del sector:** Empresas como Ericsson [13] y Nokia-Siemens [14] ofrecen plataformas de entrega de servicios (SDP), altamente robustas y basadas en la especificación JAIN SLEE, sin embargo estas soluciones son demasiado costosas y se encuentran fuera del alcance de la mayoría de las medianas y pequeñas empresas del sector. Por lo tanto este trabajo presenta una alternativa de bajo costo y altamente confiable para el desarrollo, composición y despliegue de servicios de valor agregado.

- **Publicación de artículo:** Dentro del presente trabajo de grado, se realizó la publicación del artículo *Entornos de Libre Distribución para Composición de Servicios JAIN SLEE* [15], en el marco del *Quinto Seminario Nacional de Tecnologías Emergentes en Telecomunicaciones y Telemática* [16], llevado a cabo del 10 al 12 de Junio de 2010 en la ciudad de Popayán. Este artículo presenta un análisis y evaluación de los diferentes entornos de desarrollo y composición de servicios JAIN SLEE de libre distribución que actualmente se encuentran disponibles.

1.6 Contenido

El presente trabajo de grado contiene los siguientes capítulos:

Capítulo 2: En este capítulo se describen y se analizan las diferentes tecnologías y conceptos más relevantes, que se encuentran relacionados con el desarrollo y composición de servicios de telecomunicaciones en entornos JAIN SLEE.

Capítulo 3: En este capítulo se realiza un análisis y evaluación de los diferentes entornos de desarrollo y composición de servicios JAIN SLEE de libre distribución, actualmente disponibles. Para esto se plantean una serie de criterios de evaluación, a través de los cuales es posible evaluar las diferentes herramientas.

Capítulo 4: Este capítulo refleja la contribución más importante del presente trabajo de grado. Es aquí donde se definen los diferentes pasos y lineamientos a seguir que permiten realizar composición de servicios. En este capítulo se utilizan los principios y conceptos SOA, así como también los conceptos definidos en la especificación JAIN SLEE que facilitan la composición. Finalmente, utilizando los resultados obtenidos en la evaluación realizada en el capítulo 3, se realiza la definición de los lineamientos para composición de servicios.

Capítulo 5: Este capítulo presenta el desarrollo de un prototipo que permite validar los lineamientos definidos en el capítulo 4. El prototipo consiste en la implementación de dos servicios donde se evidencia la composición y el uso de tecnologías de las telecomunicaciones y de la información. Los servicios son llamados *Numero Portable y Llamada Recordatoria.* El servicio *Numero Portable* consiste en brindar a un usuario la posibilidad de tener un número único en el cual puede ser ubicado, sin importar el número en el cual se encuentre disponible. El servicio *Llamada Recordatoria* consiste en que a través de una interfaz Web, un usuario entrega una lista de números a los cuales desea enviar un mensaje y el sistema automáticamente realiza las llamadas. Finalmente, en este capítulo se realizan unas pruebas que permiten determinar el desempeño de los servicios.

Capitulo 6: Finalmente, en este capítulo se presentan las diferentes conclusiones a las cuales se llegaron en el desarrollo del presente trabajo de grado y las diferentes ideas propuestas para la realización de trabajos futuros.

Capítulo II

En este capítulo se presenta una descripción de los conceptos y tecnologías más relevantes para el presente trabajo de grado. Específicamente, se realiza una descripción de conceptos como SOA, JAIN SLEE, SDP y de tendencias como la composición de servicios. Finalmente se describen los trabajos relacionados con la composición de servicios de telecomunicaciones.

ESTADO ACTUAL DEL CONOCIMIENTO

2.1 Contexto General

La convergencia de las redes fijas y móviles, a través de las redes de nueva generación permite la fácil creación y despliegue de nuevos servicios, en contraste con lo que ofrecían las redes tradicionales, en las que el desarrollo de los servicios estaba limitado a la infraestructura de las tecnologías de redes de acceso. Desde hace unos años, las compañías de telecomunicaciones han venido migrando sus redes de telefonía conmutada a redes de nueva generación, con el fin de enfrentar el reto de soportar los servicios tradicionales como acceso a internet, llamadas de voz, mensajería instantánea, SMS y los nuevos servicios convergentes de comunicaciones e información. Dentro de los principales objetivos al migrar a redes de nueva generación para las compañías de telecomunicaciones, se encuentra el poder ofrecer a sus potenciales clientes, servicios innovadores a unos costos eficientes [17].

La industria de las telecomunicaciones solía ser responsable por el desarrollo y despliegue de servicios, donde las organizaciones controlaban efectivamente como, donde y cuando los nuevos servicios eran creados y qué tipo de tecnología se utilizaba. Los servicios eran estrictamente propietarios, dependientes de la tecnología de la organización, manejados y entendidos por un pequeño grupo de personal calificado. Este ineficiente método de desarrollo de servicios se consideraba satisfactorio ya que el modelo de negocio centraba sus esfuerzos en captar un gran número de suscriptores y no en ofrecer una amplia gama de servicios [18].

En la actualidad los usuarios demandan servicios que satisfagan sus necesidades y que se adapten a las dinámicas condiciones del mercado. Esto requiere que las compañías del sector desarrollen nuevos servicios de valor agregado para atraer nuevos clientes, manteniendo los actuales y en consecuencia incrementar las ganancias. Por lo tanto, los operadores de telecomunicaciones necesitan tener la capacidad de desplegar rápidamente una gran selección de nuevos servicios de valor agregado o de proveer los medios para que otros desarrolladores tengan la capacidad de hacerlo, de una forma que implique bajo tiempo de desarrollo y costos reducidos.

Estos requerimientos son cumplidos por las SDP (Service Delivery Platforms), cuya arquitectura se encuentra basada en los principios de SOA (Service Oriented Architecture), los cuales redefinen la forma en la que los procesos de negocio son automatizados e implementados a través de servicios. El objetivo principal de las SDP es proveer un entorno altamente automatizado que proporcione una plataforma única para creación, aprovisionamiento, despliegue y gestión de servicios. Las SDP facilitan las

complejidades de los tipos de redes de acceso a los desarrolladores de servicios, haciendo que servicios de voz, video y contenido puedan ser combinados en servicios más complejos [18]. Esta característica hace referencia a la composición de servicios ya que todos los servicios desarrollados a través de las SDP pueden ser utilizados para construir servicios más complejos y de valor agregado, proporcionando una gran ventaja para los desarrolladores de servicios.

A continuación se exponen los conceptos más relevantes respecto a la composición de servicios y los entornos en donde es posible llevar a cabo dicho procedimiento.

2.1.1 Arquitectura Orientada a Servicios (SOA)

2.1.1.1 Introducción

En los últimos años el área de las tecnologías de la información y comunicación, ha tenido grandes cambios, surgiendo nuevas necesidades y tendencias de desarrollo aplicaciones, exigiendo así a las organizaciones imprescindibles cambios en sus modelos de negocio.

Actualmente esta situación ha conllevado a las organizaciones a plantearse nuevos objetivos que buscan poder interconectar los procesos, personas e información tanto con la propia organización como con subsidiarias y socios comerciales, dado que la falta de integración entre los componentes de IT (Tecnologías de la Información) como sistemas, aplicaciones y datos, hace difícil obtener una respuesta rápida y efectiva ante los cambios que afectan de forma natural a los negocios. La inflexibilidad genera costes, reduce la capacidad de respuesta ante los clientes y compromete el cumplimiento con las normativas legales, imponiendo una barrera que impide a las organizaciones aumentar su competitividad y crecimiento en el mercado.

Una de las soluciones propuestas para dicha problemática se llama SOA (Arquitectura Orientada a servicios). SOA establece un marco de referencia para la integración de aplicaciones independientes, donde sus capacidades y funcionalidades son ofrecidas a través de servicios. La forma más habitual de implementarla es a través de servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular [19].

Un servicio Web es un modulo que brinda una funcionalidad concreta, el cual, a través de una descripción, define sus capacidades y la manera de interactuar con él. También los servicios Web pueden componerse dentro de una aplicación completa que proporcione servicios más complejos. La estrategia de orientación a servicios permite que las aplicaciones compuestas puedan crearse con independencia de las tecnologías subyacentes como plataformas y lenguajes, facilitando así el rehúso para la creación nuevas aplicaciones y servicios [19].

Los servicios Web generalmente se basan en un conjunto de estándares de comunicación, como son XML para la representación de datos, SOAP [20] (*Simple Object Access Protocol*) para el intercambio de datos y el lenguaje WSDL [21] (*Web Services Description Language*) para describir las funcionalidades de un servicio Web. Existen más especificaciones, que definen distintas funcionalidades para el descubrimiento de servicios Web, gestión de eventos, archivos adjuntos, seguridad, gestión y fiabilidad en el intercambio de mensajes y transacciones [19].

2.1.1.2 Arquitectura de Servicios

Para la implementación de SOA, las empresas necesitan una arquitectura de servicios. Un ejemplo de dicha arquitectura se observa en la siguiente figura:

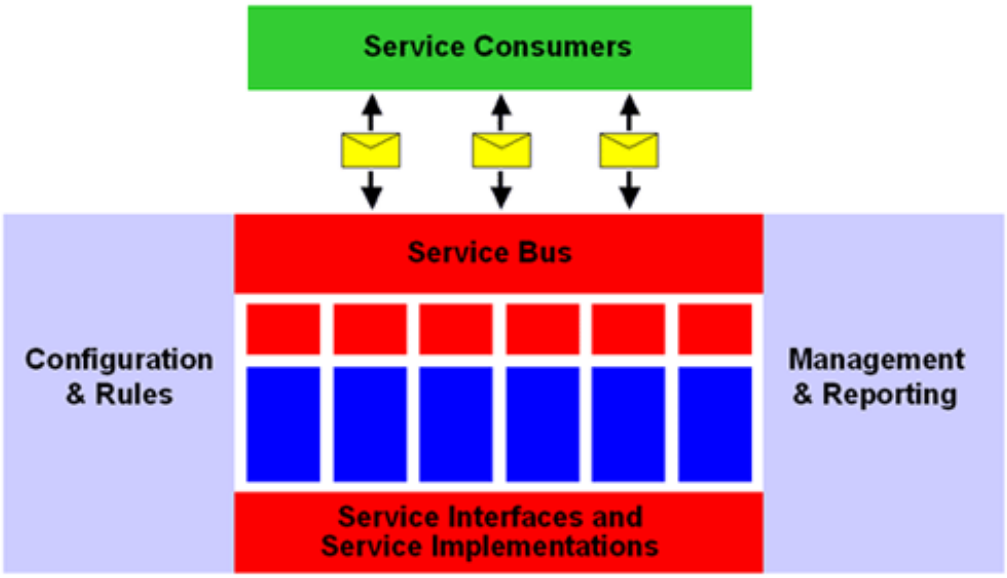

Figura 1. *Ejemplo de Arquitectura de Servicios [22].*

En la figura 1 se observa como diversos consumidores de servicios pueden invocar los servicios a través del envío de mensajes. Estos mensajes son transformados y enrutados por un *Bus de Servicios* hacia la implementación del servicio que les corresponde. La arquitectura de servicios puede proporcionar un motor reglas de negocio que permite incorporar diferentes directrices o configuraciones en uno o varios servicios. La arquitectura de servicios proporciona también una infraestructura de gestión que administra servicios y actividades como tarifación y registro de usuarios. Adicionalmente, esta arquitectura ofrece a las empresas procesos de negocio flexibles donde es posible realizar cambios a servicios individuales sin afectar otros servicios [22].

2.1.1.3 Principios de SOA

A continuación se describen una serie de principios que definen los lineamientos básicos para el desarrollo, mantenimiento y utilización de SOA.

- **Los servicios deben ser reusables:** Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo [23-24].

- **Los servicios deben proporcionar un contrato formal:** Todo servicio desarrollado debe proporcionar un contrato formal en el cual se defina: el nombre del servicio, su forma de acceso, las funcionalidades que ofrece, los datos de entrada y los datos de salida. De esta manera, todo consumidor del servicio podrá acceder a él mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio [23-24].

- **Los servicios deben tener bajo acoplamiento:** Los servicios deben ser independientes entre sí. Esto se logra a través de la utilización del contrato formal del servicio para acceder a sus diferentes funcionalidades [23-24].

- **Los servicios deben permitir la composición:** Todo servicio debe ser construido de forma que pueda ser utilizado para construir servicios más complejos [23-24].

- **Los servicios no deben tener estado:** Un servicio no debe guardar ningún tipo de información debido a que un servicio más complejo que se encuentre compuesto de otros servicios, puede sufrir de problemas de inconsistencia de datos. La información que requiera persistencia debe ser almacenada en un sistema de información independiente [23-24].

- **Los servicios deben poder ser descubiertos:** Todo servicio debe poder ser descubierto para que pueda ser utilizado. Para los servicios Web, el descubrimiento se logra a través de la publicación de sus interfaces WSDL en registros UDDI (Universal Description, Discovery and Integration) [23-25].

2.1.2 Especificación JAIN SLEE

2.1.2.1 Introducción

La especificación JAIN SLEE (Java APIs for Integrated Networks Service Logic Execution Environment) define un entorno estándar para la ejecución de lógica de servicios y especifica la manera en la cual, servicios de telecomunicaciones portables y de alta calidad, pueden ser construidos, gestionados y ejecutados. La especificación JAIN SLEE ha sido diseñada para soportar servicios de telecomunicaciones, cumpliendo con los requerimientos que exigen este tipo de servicios como: baja latencia, alta disponibilidad, orientación a eventos, etc [26].

JAIN SLEE provee un modelo de programación estándar que puede ser usado por la amplia comunidad de desarrolladores Java. El modelo de programación ha sido diseñado para simplificar el trabajo de aplicación del desarrollador, eliminar los errores de programación y asegurar que servicios altamente robustos puedan ser desarrollados en poco tiempo. JAIN es tecnología Java por lo tanto la amplia familia de API (Application Programming Interface) Java puede ser aprovechada para el enriquecimiento de las aplicaciones y servicios desarrollados bajo la especificación JAIN SLEE [26].

La especificación JAIN SLEE define una arquitectura en varias capas como se observa en la figura 2, la cual se compone de una capa central conocida como el modelo de componentes en donde se especifica como la lógica del servicio debe ser construida, empaquetada, como debe procesar eventos y como debe ser ejecutada [26].

La capa de gestión del SLEE especifica el mecanismo por el cual un administrador gestiona el SLEE (incluyendo subscripción, instalación de servicios, ciclo de vida del sistema, etc.) y permite a los desarrolladores de servicios definir los datos necesarios para un servicio en particular. Define también la gestión de especificaciones de perfil de servicios y de usuarios pertenecientes al SLEE [26].

La capa del sistema de componentes del SLEE define el enrutamiento de eventos tanto externos como eventos originados desde el SLEE y proporciona componentes comunes a todo el SLEE que pueden ser utilizados por los desarrolladores de servicios como herramientas de funcionamiento para las aplicaciones [26].

Los adaptadores de recursos son responsables de la comunicación con un recurso particular que es externo al modelo de programación de JAIN SLEE, comunicándolo con la lógica de enrutamiento de eventos en la implementación del SLEE y proporcionando al desarrollador de servicios las interfaces necesarias para hacer uso de las diferentes capacidades que proporciona una adaptador en particular [26].

La arquitectura estandarizada de los adaptadores de recursos y de las API relevantes es definida por primera vez en la especificación de JAIN SLEE 1.1. Ejemplos de recursos que son de interés para los proveedores y para los desarrolladores de servicios son: SIP, HTTP, SS7, MGCP, etc [26].

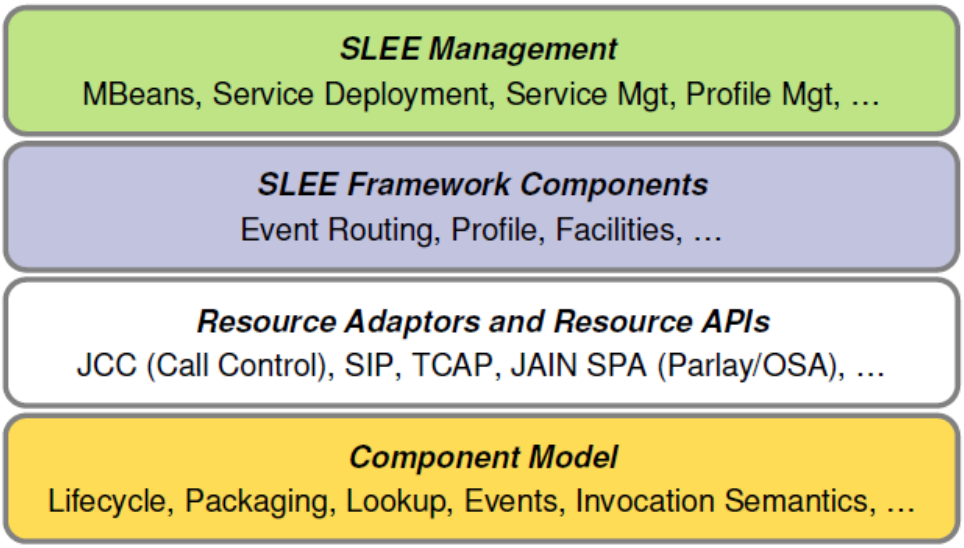

Figura 2. *Arquitectura en Capas de JAIN SLEE (Adaptación de [26]).*

2.1.2.2 Conceptos Fundamentales de JAIN SLEE

- *Perfil:* Un perfil de JAIN SLEE contiene información acerca de un servicio o de un subscriptor. Los bloques de construcción de servicio que se ejecutan dentro de un entorno JAIN SLEE pueden acceder a la información contenida dentro de los perfiles como parte de su lógica de aplicación [27].

- *Bloque de Construcción de Servicio:* El elemento de reutilización definido por JAIN SLEE es el bloque de construcción de servicio (SBB). Un SBB es un componente software que envía y recibe eventos, y también ejecuta lógica computacional basada en la recepción de eventos y su estado actual. El código de programación de un SBB está compuesto de clases Java [27].

- *Evento:* Un evento representa un acontecimiento que puede requerir procesamiento por parte de alguna aplicación. Un evento se puede originar de

diferentes fuentes, por ejemplo, una pila de protocolos externa o desde el interior del SLEE [27].

- *Recursos y Adaptadores de Recursos:* Los recursos son entidades externas que interactúan con otros sistemas fuera del SLEE, tal como elementos de red (HLR, MSC, etc.), pilas de protocolos, directorios y bases de datos. Un adaptador de recursos implementa la interfaz de un recurso dentro del entorno JAIN SLEE [26].

2.1.2.3 Características de JAIN SLEE

Las soluciones basadas en la especificación JAIN SLEE tienen las siguientes características:

- *Servicios Portables:* JAIN SLEE implementa la filosofía WORA™ (Write Once, Run Anywhere) del lenguaje de programación Java. Los componentes de aplicación pueden ser desarrollados y después ser desplegados en plataformas JAIN SLEE de diferentes proveedores sin la necesidad de recompilación o modificación del código fuente [26].

- *Consistencia, Flexibilidad y Modelo de Eventos Dinámico:* JAIN SLEE provee un fuerte soporte para las aplicaciones asíncronas con un modelo de eventos dinámico, consistente y flexible. Los componentes de las aplicaciones reciben eventos de los canales de eventos que son establecidos en tiempo de ejecución. Los recursos de red (tal como las pilas de protocolos) crean representaciones de las "llamadas" y dirigen los eventos generados por las mismas hacia el SLEE. De esta manera los componentes de las aplicaciones son invocados por el SLEE para procesar esos eventos en un contexto transaccional [26].

- *Arquitectura de Componentes Orientada a Objetos:* JAIN SLEE es la arquitectura de componentes estándar para La construcción de aplicaciones de comunicaciones distribuidas y orientadas a objetos en el lenguaje de programación Java. La especificación JAIN SLEE define un modelo de componentes para estructurar la lógica de aplicación de las aplicaciones de comunicaciones como una colección de componentes reutilizables orientados a objetos, y para la composición de esos componentes en servicios más complejos de valor agregado. La especificación también define el contrato entre los componentes y el contenedor que albergará estos componentes durante el tiempo de ejecución [26].

- *Independencia de la Red:* El modelo de programación de JAIN SLEE es independiente de cualquier protocolo, API o topología de red. Esto es soportado a través de la arquitectura del adaptador de recursos. Muchas tecnologías de red pueden ser integradas en el entorno SLEE. Por lo tanto, JAIN SLEE puede ser utilizado para solucionar problemas de negocio que involucren múltiples redes [26].

- *Soporte para Servicios de Voz y Servicios Web:* JAIN SLEE permite la interoperabilidad con J2EE, permitiendo el desarrollo de soluciones basadas en servicios convergentes de voz, datos y servicios Web [26].

- *Soporte para Integración con Sistemas de Gestión Existentes:* La especificación de JAIN SLEE define una API de gestión que permite al entorno SLEE ser controlado por un sistema de gestión externo. Se definen interfaces que soportan la instalación y mantenimiento de servicios y especificaciones de perfil [26].

2.1.2.4 Integración de JAIN SLEE en Entornos de Telecomunicaciones

Debido a que JAIN SLEE es un estándar abierto, flexible, de alto rendimiento y basado en el procesamiento de eventos, está encontrando un buen uso en un gran número de entornos de comunicaciones [26].

- JAIN SLEE puede ser usado como plataforma para el despliegue de servicios en las redes de nueva generación (NGN). Sin embargo, también soporta la integración con las redes de telecomunicaciones tradicionales (SS7), a través del uso de adaptadores de recursos.

- JAIN SLEE provee conexión a sistemas OSS/BSS (Operational Support System/Business Support System). Se pueden crear servicios para proveer eficiencia operacional a los operadores, así como servicios de gestión para los consumidores.

- JAIN SLEE provee un entrono de ejecución abierto, conectado de manera externa al entorno seguro de un operador, a través de la pasarela Parlay/OSA/JSPA.

- JAIN SLEE es capaz de interoperar con J2EE, por lo tanto servicios enriquecidos basados en el control de llamada y servicios Web pueden ser desplegados.

- Considerando sus capacidades de integración y dependiendo de las perspectivas y requerimientos, un entorno JAIN SLEE puede ser visto como un mediador o Middleware, para las telecomunicaciones.

En la figura 3 se pueden observar los diferentes escenarios de telecomunicaciones descritos anteriormente, en los cuales es posible realizar una integración con un entorno JAIN SLEE.

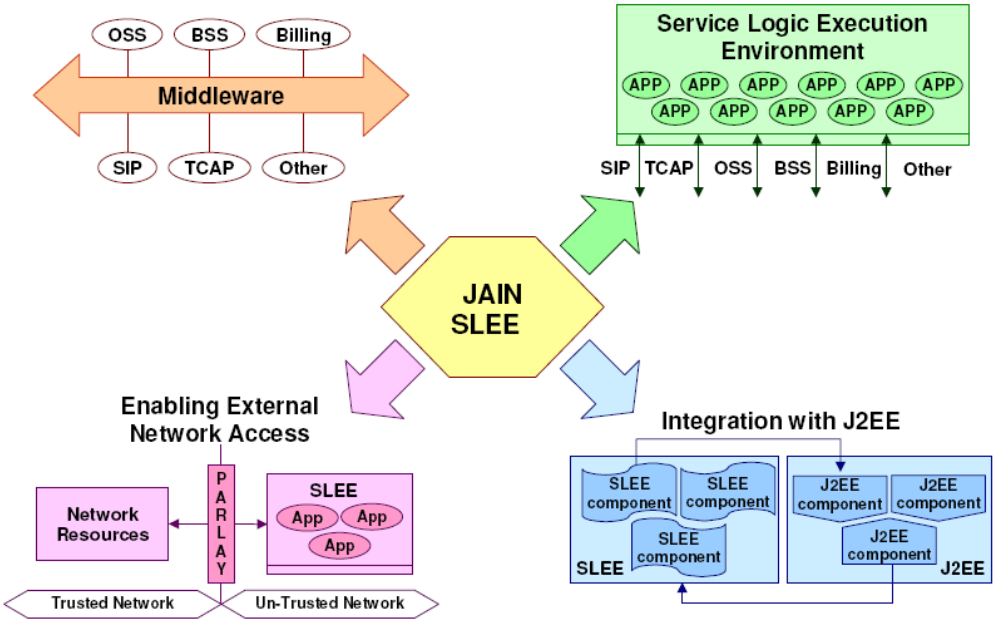

Figura 3. *JAIN SLEE en un Entorno de Telecomunicaciones* [26].

2.1.3 Plataforma de Entrega de Servicios (SDP)

El concepto de SDP (Service Delivery Platform) fue creado esencialmente para describir una solución para la entrega de servicios de valor agregado a consumidores y empresas. Dicha solución puede ser desplegada mediante proveedores de servicios de redes fijas y móviles. De esta manera una plataforma de entrega de servicios SDP tiene como objetivo principal permitir a los proveedores el rápido desarrollo y despliegue de nuevos servicios convergentes de valor agregado [18].

La SDP es un modelo basado en estándares, que generalmente proporciona un entorno de creación y composición de servicios, un entorno de ejecución y gestión de servicios, y abstracciones para control de medios, presencia, localización y otras funciones para comunicaciones a bajo nivel con el fin de lograr una integración de los servicios de telecomunicaciones con los servicios de internet [18].

Componentes básicos de una SDP:

- *Entorno de creación de servicios (SCE):* El SCE es el principal componente para los desarrolladores de servicios de telecomunicaciones, ya que este es utilizado para crear software, scripts y demás recursos, que hacen parte de los servicios desplegados. En un SCE se lleva a cabo la construcción paso a paso de los servicios, describiendo los elementos e interacciones entre sí que conforman el ciclo de vida del servicio; además el SCE cuenta con una variedad de componentes que permiten la reutilización de servicios y por lo tanto la composición de estos, con el fin de obtener robustos procesos de negocio.

- *Entorno de ejecución de servicios (SEE):* El SEE es el componente de la plataforma que soporta la ejecución de los servicios que han sido creados a través del SCE, cumpliendo con ciertos requerimientos de escalabilidad, baja latencia y

tolerancia a fallos. Para dicha tarea este entorno usualmente contiene una Gateway de eventos y un motor de ejecución lógica, que permite realizar la composición de servicios descrita en el SCE.

- *Habilitadores:* Los Habilitadores son bloques de construcción que exponen una simple API, la cual permite al desarrollador abstraer la complejidad de las funciones de la red subyacente. Estos son los pilares del futuro desarrollo de servicios, y de la expansión del alcance de los desarrolladores dispuestos a proveer aplicaciones sobre redes de telecomunicaciones. Una SDP puede hacer uso de Habilitadores que se encuentren tanto dentro de la plataforma como fuera de ella, para lograr servicios que utilicen variedad de funciones. Usualmente se utilizan Habilitadores para control de llamadas, gestión de perfiles de usuario, presencia, localización, mensajería, control de medios, aplicaciones SIP, e integración con sistemas OSS/BSS.

Dado que las SDP deben ser implementadas basándose en estándares de la industria, actualmente existen muchas organizaciones enfocadas en proponer una arquitectura estándar para las SDP. Con base en este objetivo se presenta la tendencia de incorporar los conceptos de SOA a la implementación de las SDP y de esta manera obtener una gran capacidad de integración de las tecnologías de los operadores y demás beneficios de SOA. Por otra parte los operadores virtuales y de menor nivel podrían construir sus SDP utilizando Habilitadores estándar pre-integrados por terceros y de esta manera reducir notablemente tiempo de desarrollo de servicios, costos y riesgos de integración, brindándole a las empresas una mayor competitividad en el mercado.

Teniendo en cuenta la anterior problemática se han hecho grandes aportes por parte de algunas organizaciones. Por ejemplo, OMA [28] trabaja en políticas de control, TISPAN [29] y 3GPP [30] trabajan en la integración con arquitecturas IMS, W3C [31] e IETF [32] trabajan con aplicaciones Web, y OASIS [33] y W3C en integración con IMS (IP Multimedia Subsystem) mediante servicios Web.

Actualmente el mercado de las SDP presenta un continuo crecimiento debido a la cantidad de soluciones que están emergiendo por parte de los proveedores, los cuales aun están incursionando en el mercado. Entre estos proveedores se distinguen 2 tipos, unos son quienes proveen una arquitectura completa para la SDP, ya sea basándose en componentes propietarios o utilizando componentes estandarizados. El otro tipo implementa componentes específicos para la SDP de otro operador como por ejemplo entornos de creación de servicios, entornos de ejecución o habilitadores particulares.

Existe un número significativo de proveedores en el mercado de las SDP, entre los cuales se destacan los siguientes:

2.1.3.1 Plataforma OpenCloud Rhino

El entorno de ejecución de lógica de servicios (SLEE) Rhino es un servidor de aplicaciones que permite el desarrollo de aplicaciones de telecomunicaciones. Es una plataforma Java que soporta las especificaciones JAIN SLEE 1.0 (JSR 22) y JAIN SLEE 1.1 (JSR 240). El SLEE Rhino puede ser usado para desarrollar y desplegar aplicaciones que usen los protocolos basados en SS7 tales como INAP (Intelligent Network Application Part) y CAP (CAMEL Application Part), protocolos IMS (IP Multimedia Subsystem) como ISC y Diameter, y protocolos basados en IP como HTTP y SIP. Rhino tiene una

infraestructura dirigida a los operadores de telecomunicaciones y se caracteriza por ser tolerante a fallos, por proveer una disponibilidad continua y por soportar una gestión en línea. La plataforma Rhino provee las siguientes herramientas [34]:

- El servidor SLEE Rhino.
- Herramientas de gestión para tareas como despliegue de servicios, configuración de alarmas y modificación de perfiles.
- Integración con RDBMS (Relational Database Managment System) y servidores de directorio (los servicios pueden usar los mecanismos Java estandarizados para acceder a bases de datos relacionales y a servidores de directorio).
- Una Arquitectura de adaptadores de recursos que provee un marco de referencia en el cual los adaptadores de recursos son desarrollados y posteriormente desplegados en el SLEE Rhino. OpenCloud tiene un número de paquetes de conectividad (grupo de adaptadores de recursos) para protocolos SS7, IMS y mensajería.

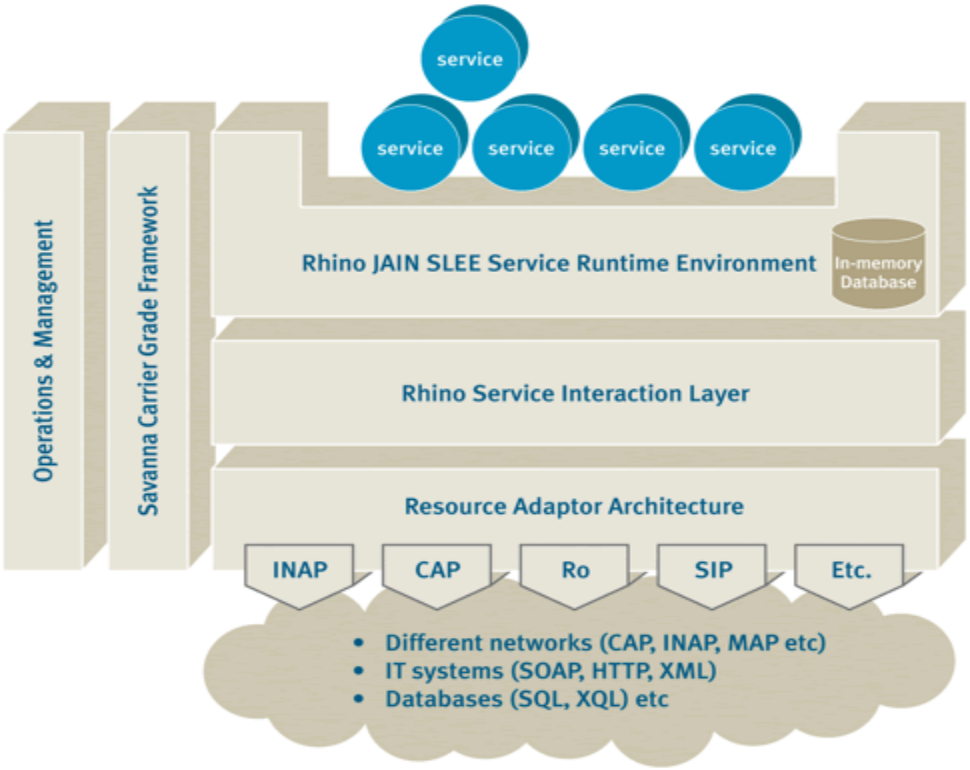

Figura 4. *Arquitectura de la Plataforma OpenCloud Rhino* [34].

La figura 4 presenta la arquitectura propuesta por Open Cloud para su plataforma Rhino, en donde es posible observar la utilización de un entorno de ejecución de servicios basado en la especificación JAIN SLEE y una arquitectura de adaptadores de recursos que implementa las interfaces de diferentes protocolos y recursos de red.

2.1.3.2 Ericsson-SDP

La SDP de Ericsson es un conjunto de componentes que permiten a los operadores entregar nuevos servicios a través de una red horizontal de servicios, la cual está

diseñada para entregar servicios sobre múltiples redes de acceso. Esta SDP tiene como objetivo principal, soportar una fuerte integración con las operaciones y procesos de negocio de los operadores, además de proporcionar la posibilidad de gestionar completamente el ciclo de vida de los servicios, incluyendo la creación, despliegue, aprovisionamiento, finalización, facturación, operación y mantenimiento.

La construcción de esta SDP está basada en el principio de reutilización de funcionalidad de SOA, por lo tanto se definieron componentes que permiten agrupar las funciones. Estos componentes son el entorno de creación de servicios (SCE), servidores de aplicaciones, los habilitadores de servicios y un componente de soporte de funciones comunes. A continuación se describen algunos de ellos [35].

- *SCE:* permite creación de servicios con implementación propia o externa, puede utilizar los diferentes habilitadores de servicio y utiliza entornos de desarrollo y herramientas estándar. También expone las funciones de red de forma simple para desarrollos internos o externos, abstrae elementos comunes de la lógica del servicio como un componente reutilizable y permite a los operadores centrarse en procesos de negocio al desarrollar aplicaciones que pueden ser integradas con otros sistemas informáticos.

- *Servidor de aplicaciones:* plataforma de aplicaciones para unificar desarrollo, ejecución, gestión y la lógica de negocio del operador.

- *Soporte de funciones comunes:*

    - Aprovisionamiento: provee datos acerca del servicio y el usuario.

    - Gestión de dispositivos: proporciona facilidad de uso de nuevos servicios para el usuario.

    - Catalogo de servicios: provee un registro de todos los servicios teniendo en cuenta sus características de implementación.

    - Acceso a datos: provee un perfil de usuario con sus servicios y características

    - Facturación: factura todos los servicios de usuario en tiempo real.

    - Gestión de identidad: gestiona la identidad digital del usuario dentro de la red.

    - Operación y mantenimiento: gestión configuración y rendimiento para los sistemas y aplicaciones del servicio.

    - Integración de habilitadores de servicio: provee soporte para procesos de negocio, exponiendo los habilitadores de servicio a desarrolladores de aplicaciones externos, y a los servidores de aplicaciones.

En la figura 5 se presenta la arquitectura de la SDP de Ericsson y se observan los componentes anteriormente descritos.

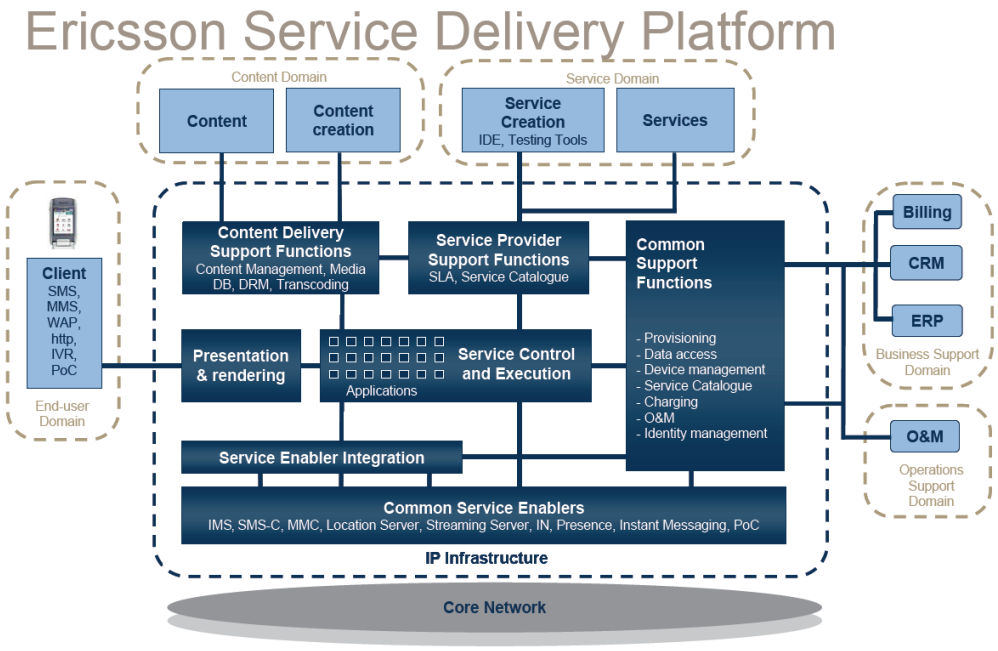

Figura 5. *Arquitectura de Ericsson SDP* [35].

2.1.3.3 Plataforma de Comunicaciones Mobicents

La plataforma de comunicaciones Mobicents [7] es un altamente escalable servidor de aplicaciones de libre distribución, orientado a eventos, con un robusto modelo de componentes y un entorno de ejecución tolerante a fallos. En este proyecto participan desarrolladores pertenecientes a organizaciones como Portugal Telecom Inovação, Telecom Italia, Universidad de Genova, Vodafone R&D, T-Mobile, Lucent Technologies, Open Cloud, Aepona, NEC Japan, JBoss y Red Hat.

La arquitectura de la plataforma de comunicaciones Mobicents se encuentra diseñada para la creación, despliegue y gestión de servicios y aplicaciones que integran voz, video y datos a través de redes IP y de comunicaciones. La plataforma de comunicaciones Mobicents logra la convergencia aprovechando las siguientes capacidades centrales [36].

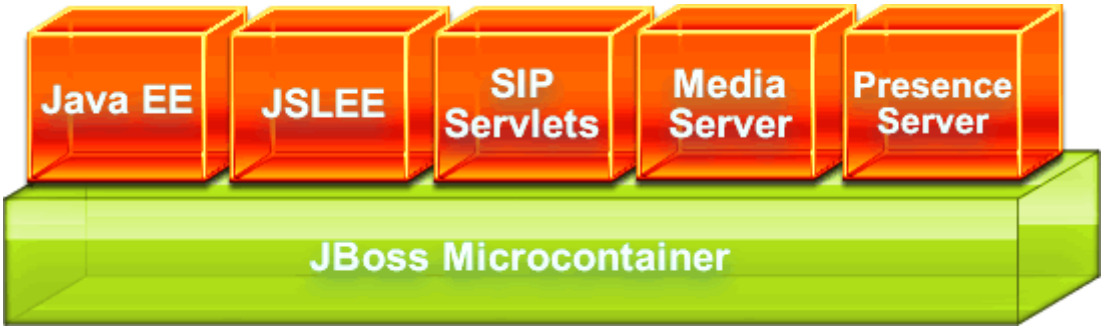

Figura 6. *Capacidades del Servidor de Aplicaciones Mobicents* [7].

- *JAIN SLEE:* Mobicents JAIN SLEE es una implementación certificada de la especificación JAIN SLEE v1.1 (JSR 240) de libre distribución. Mobicents brinda un modelo de componentes y un entorno de ejecución robusto para aplicaciones

de telecomunicaciones. Es un complemento para J2EE para permitir convergencia de voz, video y datos en aplicaciones inteligentes de nueva generación [36].

- *Sip Servlets:* Mobicents Sip Servlets entrega una plataforma abierta y consistente en la cual desarrollar y desplegar servicios SIP y JEE portables y convergentes. Es la primera implementación certificada de la especificación SIP Servlet v1.1 (JSR 289) de libre distribución que trabaja sobre contenedores Tomcat y JBoss. Se caracteriza por proveer ejecuciones eficientes y seguras, y por fomentar la innovación y el desarrollo de aplicaciones interoperables entre Sip Servlets y JAIN SLEE para que de esta forma se logre explotar las fortalezas de ambas especificaciones [36].

- *Media Server:* Mobicents Media Server es un servidor de libre distribución basado en la implementación Java Media Framework cuyo objetivo principal es proveer una funcionalidad completa y competitiva de una pasarela de contenido de la más alta calidad. El servidor satisface las necesidades de redes inalámbricas, cableadas y de redes fijas y móviles IP convergentes desde una única plataforma de contenido. También incrementa la flexibilidad con una pasarela de contenido que soporta una amplia variedad de protocolos de control de llamada, los cuales satisfacen necesidades empresariales y de pequeños proveedores [36].

- *Presence Server:* Mobicents Presence Server es la primera implementación de libre distribución de un servidor de gestión de documentos XML (XDM), como es definido por la Open Mobile Alliance (OMA) en la especificación Presence XDM Specification. Este elemento funcional de las redes IP de nueva generación es responsable de la gestión de los documentos XML de los usuarios, tales como reglas de autorización de presencia, listas de contactos y de grupos, información de presencia estática, etc [36].

- *JBoss Microcontainer:* JBoss Microcontainer es el entorno de almacenamiento en el cual residen los contenedores de más alto nivel. Provee servicios de registro, configuración, gestión, control de utilización de clases, empaquetamiento, despliegue y muchos otros bloques necesarios en servidores altamente escalables y tolerantes a fallos [36].

En el ámbito de las telecomunicaciones y de las redes de nueva generación, Mobicents se presenta como un núcleo de alto rendimiento para plataformas de entrega de servicios (SDP). Mobicents permite la composición de bloques constructores de servicio (SBB) tales como control de llamada, facturación, aprovisionamiento de usuario, administración y capacidades de detección de presencia [36].

La especificación JAIN SLEE garantiza, que pilas de protocolos ampliamente difundidas como SIP, sean utilizadas como adaptadores de recursos. Los SBB de JAIN SLEE tienen grandes similitudes con los EJB, y naturalmente es posible llevar a cabo su integración con aplicaciones empresariales, aplicaciones Web, aplicaciones de gestión y terminales SOA. El monitoreo y la gestión de los componentes de Mobicents es lograda a través de aplicaciones JMX (Java Managment Extensions) y de interfaces SNMP (Simple Network Managment Profiles) [36].

Más allá de las telecomunicaciones, Mobicents es aplicable a una amplia variedad de problemas que demandas señalización de alto volumen y baja latencia. Por ejemplo,

transacciones financieras, juegos en línea, integración de redes con sensores RFID (Radio Frequency IDentification) y control distribuido [36].

2.2 Trabajos Relacionados

2.2.1 Composición de Servicios

En la industria se han utilizado diferentes términos para describir como los componentes pueden ser conectados entre sí para construir complejos procesos de negocio. Los sistemas de descripción del flujo de trabajo han existido como un medio para manejar el enrutamiento de procesos entre varios recursos en una organización de comunicaciones o de tecnologías de información. Estos recursos pueden incluir personas, sistemas o aplicaciones que típicamente involucran alguna intervención humana. Los sistemas de administración de procesos de negocio (BPMS) se han utilizado también para permitir que se construyan modelos de diseño de procesos con varias actividades de integración (por ejemplo integración con sistemas legados). Generalmente, los sistemas BPMS pueden cubrir completamente el ciclo de vida de un proceso de negocio, incluyendo las tareas de modelado, ejecución, monitoreo, administración y optimización [37].

Con la introducción de los servicios Web, términos como composición de servicios y flujo de servicios, son utilizados para describir la composición de los mismos en un proceso de negocio. Más recientemente, los términos de orquestación y coreografía se han utilizado para describir esta tendencia. La orquestación describe cómo los servicios pueden interactuar entre sí a nivel de mensajes, incluyendo la lógica del negocio y el orden de ejecución de las interacciones. Estas interacciones pueden involucrar aplicaciones y organizaciones, resultando en un modelo transaccional de procesos de múltiples etapas. La coreografía lleva registro de la secuencia de mensajes que involucran múltiples actores y múltiples fuentes, incluyendo clientes, proveedores y socios. Típicamente, la coreografía es asociada con el intercambio publico de mensajes que ocurre entre múltiples servicios, en lugar de un proceso de negocio especifico que es ejecutado por un único actor. Existe una importante distinción entre orquestación de servicios y coreografía de servicios. La orquestación hace referencia a un proceso de negocio ejecutable que puede interactuar tanto con servicios internos como externos al entorno de ejecución. Para la orquestación, el proceso es siempre controlado desde la perspectiva de uno de los actores del negocio. La coreografía es más colaborativa por naturaleza, en la cual cada actor involucrado en el proceso describe el rol que realiza en la interacción. Sin embargo, las mejoras recientes y el desarrollo de estándares convergentes, han causado que esta distinción no sea tan visible. Por lo tanto, actualmente el termino de orquestación de servicios se utiliza para describir la creación de procesos de negocio, ya sean ejecutables o colaborativos [37].

Existe un número importante de requerimientos técnicos que deben ser tenidos en cuenta cuando se diseñan procesos de negocio que involucran servicios de telecomunicaciones y de información. El conocimiento de estos requerimientos ayudará en el correcto posicionamiento de los diversos estándares y plataformas que han sido introducidos para la composición de servicios. La capacidad de invocar servicios de forma asíncrona es vital para alcanzar la confiabilidad, escalabilidad y la adaptabilidad requerida por los entornos actuales de comunicaciones e información. La arquitectura debe tener en cuenta cómo responderá el sistema, si ocurre un error o si el servicio o servicios invocados no responden en un tiempo dado.

La especificación JAIN SLEE cumple con los requisitos técnicos previamente descritos para la composición de servicios de comunicaciones y de información. En esta especificación, la composición de servicios se realiza a través de los bloques constructores de servicio (SBB) definidos por la misma. En un entorno JAIN SLEE el desarrollador del servicio define las diferentes relaciones e interacciones que deben sostener los SBB que componen un servicio en tiempo de ejecución.

2.2.2 Composición e Integración de Servicios Web de Comunicaciones

En este artículo se propone la creación de nuevos servicios de telecomunicaciones de valor agregado, basándose en la integración entre los servicios ofrecidos por los proveedores de tecnologías de la información, y los ofrecidos por operadores de telecomunicaciones. Dicha tarea presenta cierta dificultad dado que a pesar de que en el artículo se reconoce a los servicios Web como la mejor opción para la provisión, composición y realización de servicios de telecomunicaciones, el modelo tradicional de implementación de servicios Web no cumple satisfactoriamente los requerimientos de las telecomunicaciones tales como baja latencia, interacciones asíncronas, tiempo real y eventos de granularidad fina y alta frecuencia. Por lo tanto en el artículo se plantea la necesidad de una plataforma de servicios con arquitectura basado en eventos e iteraciones asíncronas.

Para solucionar dicha necesidad se propone la construcción de una plataforma basada en la especificación JAIN SLEE, la cual define una nueva forma de crear servicios de valor agregado que cumplan con los requerimientos de las telecomunicaciones. Esta plataforma consiste en 2 módulos, el primero es un servidor de aplicaciones llamado StarSLEE, el cual es capaz de ejecutar los componentes de JAIN SLEE definidos como SBB y proveer el soporte y recursos necesarios por el servicio. El segundo modulo es el StarSCE, un entorno de creación de servicios que provee una gran variedad de utilidades para crear servicios de valor agregado para telecomunicaciones [38].

Posteriormente en el artículo se presenta una descripción teórica de las características y funciones de la plataforma, enfocándose en resaltar las ventajas que posee. Esta descripción abarca la arquitectura y modelo de componentes, el proceso de composición de servicios, y el mecanismo de descubrimiento de servicios. Además cabe resaltar que en la descripción y desarrollo del artículo se enfatiza en la convergencia de aplicaciones Web y aplicaciones de comunicaciones, por lo que se resalta la importancia de la integración de protocolos como SIP y SOAP, con base en esto describe los procesos que se llevan a cabo en la plataforma mediante ejemplos de dicha integración. Finalmente propone un método para desplegar una implementación de un servicio de comunicación basado en JAIN SLEE, como un servicio Web tradicional, pero que cumple con los requerimientos de las telecomunicaciones [38].

Entre las características más importantes de este trabajo se destaca la implementación de la especificación JAIN SLEE, mediante el servidor de Aplicaciones de Comunicaciones StarSLEE y el entorno de Creación de Servicios StarSCE. Además otras características como la fácil creación de servicios mediante una interfaz grafica, la disposición de una variedad de adaptadores de recursos que permite una total interoperabilidad, y el Proxy UDDI que mejora notablemente el descubrimiento de servicios mediante personalización en la búsqueda [38].

Se consideran como aportes de este proyecto, toda la base teórica planteada, ya que permite conocer de forma sencilla, acerca de la especificación JAIN SLEE y su aplicación en una plataforma de servicios. Además aporta valiosas propuestas acerca de mecanismos de descubrimiento de servicios, que posteriormente se podrían integrar con procesos de búsqueda semántica.

Como una gran desventaja, se reconoce que aunque la implementación desarrollada es muy completa y cumple satisfactoriamente los requisitos para el desarrollo y composición de servicios de telecomunicaciones de valor agregado, no se encuentra disponible para uso académico ni comercial, además, no existe documentación detallada de libre acceso del funcionamiento e implementación del sistema, luego no es una opción viable para los desarrolladores interesados en la creación de nuevos servicios de valor agregado. Tampoco ofrece una guía o directrices que describan los pasos que se deben seguir para realizar composición de servicios de telecomunicaciones en entornos JAIN SLEE.

2.2.3 Proyecto SPICE

El proyecto SPICE [39] (Service Platform Innovative Communication Enviroment), tiene por objetivo principal, promover la diseminación de servicios móviles personalizados y eficientes, permitiendo que los desarrolladores puedan definir servicios enriquecidos semánticamente. Dentro de este proyecto se definió un lenguaje de programación llamado SPATEL [40], el cual permite la definición de servicios ejecutables semánticamente enriquecidos.

En el proyecto SPICE se definen dos entornos principales:

- Un entorno de creación de servicios (SCE), el cual contiene herramientas para los desarrolladores que permiten la creación y composición de servicios.

- Un entorno de ejecución de servicios (SEE), el cual incluye un servidor de aplicaciones, donde se despliegan los servicios y también dispone de una infraestructura de nivel inferior.

La Universidad de Twente contribuyó en el proyecto con el desarrollo de un motor de composición automática (ACE) [41], el cual es el componente clave del SCE. El objetivo principal del ACE es permitir la composición dinámica de servicios. Uno de los bloques de construcción principales del ACE es el algoritmo de composición, el cual automáticamente realiza la composición de servicios a partir de servicios previamente desarrollados y desplegados en un entorno de comunicaciones, de acuerdo a una petición semántica especifica realizada por un usuario final.

2.2.3.1 La Arquitectura SPICE

El objetivo principal del proyecto SPICE es facilitar el desarrollo despliegue y ejecución de servicios para entornos móviles. El entorno de creación de servicios (SCE) permite los procesos de desarrollo, mientras que el entorno de ejecución de servicios (SEE) es responsable del despliegue y ejecución de los servicios. Ambos entornos se componen de diferentes herramientas y su arquitectura está altamente inspirada por la Arquitectura Dirigida por Modelos (MDA). Siguiendo los fundamentos de MDA, el SCE está principalmente compuesto de herramientas que trabajan con modelos específicos,

mientras que el SEE trabaja con componentes desplegables y ejecutables que son modelados en el SCE [42].

Un desarrollador de servicios puede crear componentes básicos, donde cada componente modela un servicio individual. EL modelo de cada servicio es producido usando el lenguaje SPATEL. Un componente básico contiene la implementación de un servicio en la forma de un componente ejecutable en el SEE. El desarrollador de servicios puede definir también la composición de servicios, la cual consiste en componentes básicos de servicios relacionados entre sí. Una composición de servicios genera un archivo descriptor, el cual es ejecutable en el SEE [42].

Un usuario final puede realizar peticiones para crear sus propias composiciones de servicios basadas en servicios existentes. Las peticiones de composición deben ser realizadas en cualquier lenguaje amigable para el usuario incluyendo lenguaje natural, las cuales son transformadas a SPATEL para poder ser procesadas [42].

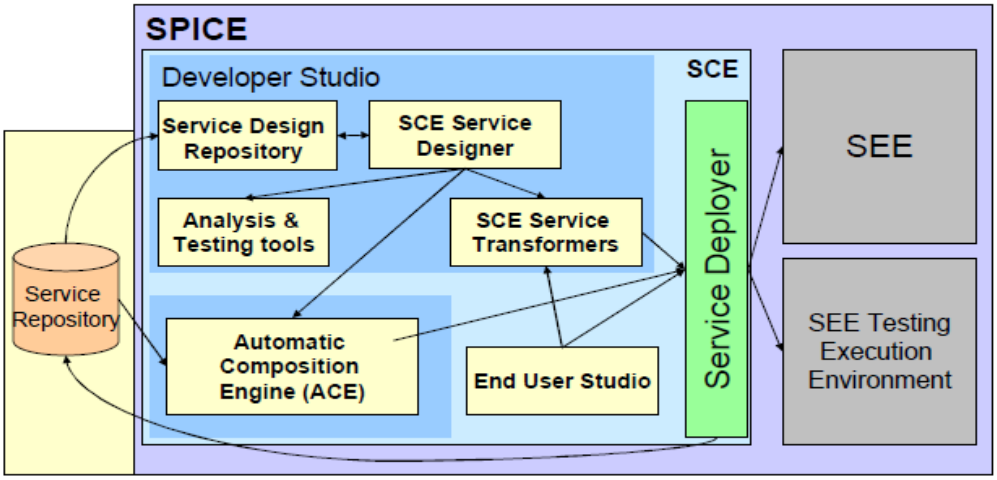

Figura 7. *Arquitectura del Proyecto SPICE* [42].

La figura muestra los componentes del SCE, donde el estudio del desarrollador y el estudio del usuario final proveen las herramientas de desarrollo para la creación, despliegue, ejecución y composición de servicios. El motor de composición automática soporta la composición de servicios. Un componente de despliegue de servicios, despliega los servicios en el SEE previamente creados en el SCE. El SCE se comunica con el repositorio de servicios el cual contiene las definiciones de diferentes servicios.

2.2.3.2 El Lenguaje de Descripción SPATEL

En el marco del proyecto SPICE se ha desarrollado un leguaje para descripción de servicios de telecomunicaciones SPATEL (SPice Advanced service description language for TELecommunication services), el cual es parte fundamental del entorno de creación de servicios en la arquitectura SPICE. SPATEL puede ser usado para modelar servicios, tanto “atómicos” como servicios compuestos [42].

SPATEL ofrece dos formalismos distintos: un formalismo para desarrolladores y un formalismo para usuarios finales. Se ofrece un formalismo para usuarios finales porque el proyecto SPICE quiere permitir que los usuarios finales puedan crear sus propios

servicios personalizados, basados en servicios básicos previamente existentes. Este formalismo es más restringido que el formalismo de los desarrolladores debido al bajo nivel de experiencia de los usuarios y a la necesidad de mantener limitadas sus capacidades por razones de seguridad [42].

Otro aspecto clave de SPATEL es que soporta la descripción de parámetros funcionales (objetivos, entradas, salidas, precondiciones y efectos) y propiedades no funcionales (calidad de servicio, costos, etc.) por medio del uso de anotaciones semánticas. Estas anotaciones semánticas pueden ser definidas como referencias a términos en ontologías. El proyecto SPICE provee un conjunto de ontologías que está destinado a soportar los casos de uso más comunes en un entorno de telecomunicaciones [42].

2.2.3.3 Aportes y Desventajas del Proyecto SPICE

Los principales aportes de este proyecto son los siguientes:

- La creación de mecanismos que permiten el desarrollo y evaluación de la composición dinámica de servicios usando las descripciones de servicios enriquecidos semánticamente.
- El planteamiento de un modelo para la composición dinámica de servicios de comunicaciones y de servicios de información en entornos móviles.
- La capacidad de permitir a los usuarios finales la creación de servicios personalizados a partir de la composición de servicios previamente desarrollados.

El proyecto SPICE es desarrollado por varias compañías de telecomunicaciones de la Unión Europea en conjunto con algunas universidades. Este hecho representa una gran limitante debido a que el proyecto SPICE no proporciona implementaciones de libre distribución, por lo tanto no es posible tener acceso a esta tecnología. Otro aspecto a tener en cuenta desde el punto de vista de la composición de servicios de telecomunicaciones, es que el proyecto SPICE centra su desarrollo en una plataforma de creación, composición y ejecución de servicios de telecomunicaciones para entornos móviles, lo cual plantea una limitante en cuanto a la integración de las redes de telecomunicaciones fijas y su acceso a los nuevos servicios de valor agregado basados en Web 2.0. Otro aspecto a tener en cuenta es que a pesar de que el lenguaje SPATEL permite la composición de servicios gracias al enriquecimiento semántico de los servicios, no se definen unos lineamientos que describan la utilización de este lenguaje para componer servicios.

2.2.4 Proyecto TeamCom

El proyecto TeamCom [43] consiste en una plataforma para desarrollo y despliegue de servicios sobre redes de nueva generación. Dicha plataforma tiene una arquitectura de red basada en IMS (IP Multimedia Subsystem) que facilita la integración de diferentes redes, y además controla el acceso a los servicios ofrecidos. Otra característica importante de este proyecto es el entorno de creación de servicios, el cual está basado en la especificación JAIN SLEE y el lenguaje BPEL.

Este entorno de creación de servicios sigue una nueva propuesta planteada en [44], la cual pretende desarrollar servicios mediante la orquestación de componentes JAIN SLEE, haciendo uso del lenguaje de composición de servicios BPEL. Para llevar a cabo dicho proceso, primero se diseña el servicio en BPEL basándose en la composición de 8

servicios Web predefinidos, los cuales proveen las funciones de telecomunicaciones que soporta JAIN SLEE. Posteriormente los archivos generados en BPEL son convertidos en archivos Java (SBB's, descriptores de despliegue, librerías y adaptadores de recursos) por medio de una herramienta también desarrollada en [44], los cuales implementan el servicio según la especificación JAIN SLEE. De esta manera se diseñan servicios robustos en BPEL que son desplegados como componentes JAIN SLEE.

En la figura 8 (a) es posible observar el proceso llevado a cabo para el desarrollo del servicio, y en la figura 8 (b) se muestra el entorno de creación de servicios teniendo en cuenta la interfaz y modulo de orquestación BPEL, y el SLEE.

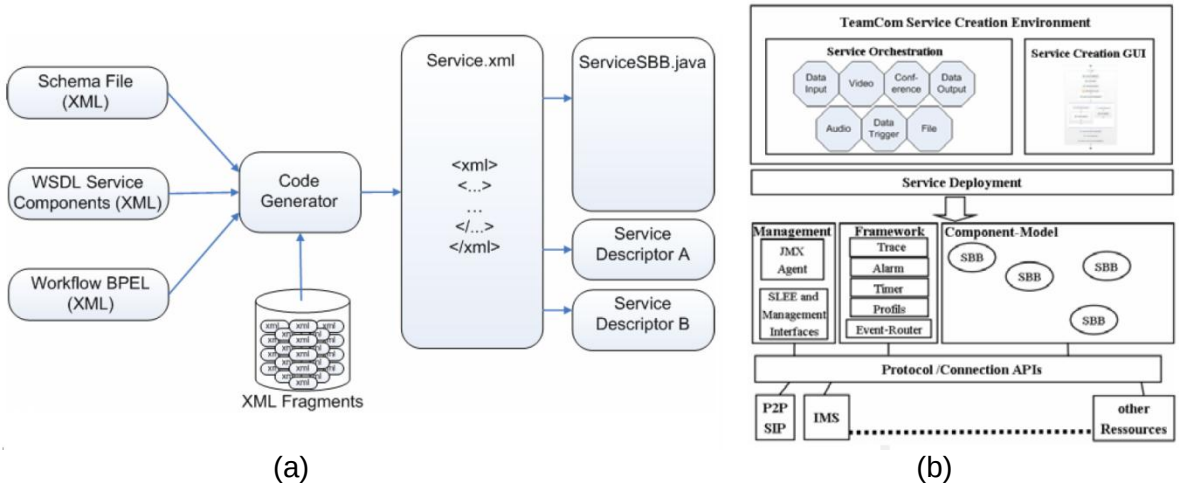

Figura 8. *(a) Creación del Servicio, (b) Entorno de Desarrollo del Proyecto TeamCom* [43]*.*

Dentro del proyecto TeamCom vale la pena resaltar la forma sencilla de desarrollar servicios convergentes JAIN SLEE, por medio de una interfaz grafica amigable al usuario, además este proyecto tiene un aporte significativo dado que el lenguaje BPEL es bastante reconocido y utilizado para desarrollo de servicios de información. Sin embargo existen algunos aspectos negativos dentro del proyecto TeamCom, tales como la no disponibilidad de la herramienta que convierte el archivo BPEL en un SBB, haciendo imposible su utilización por parte de desarrolladores interesados, además dado que para definir el servicio en BPEL, se utilizan 8 servicios Web predefinidos, solo es posible obtener una funcionalidad básica que no explota todo el potencial de JAIN SLEE.

Este proyecto ha servido como base para investigaciones posteriores, que buscan aprovechar las ventajas de la conversión BPEL a JAIN SLEE, un ejemplo específico se puede observar en [45], en donde es planteada dicha conversión de una forma similar, pero con un modelo distinto en el cual se propone incluir un motor de ejecución BPEL dentro del SLEE para realizar la composición entre servicios Web y bloques constructores de servicios (SBB's) JAIN SLEE. Para esto se implementa un adaptador de recursos BPEL, el cual aloja un contenedor Servlet que tiene desplegado el motor BPEL, y un SBB proxy que actúa como mediador entre otros SBB's y el motor BPEL. Este trabajo logra la composición entre servicios Web y SBB's, pero la adaptación del motor BPEL disminuye el rendimiento en tiempo real del SLEE, además el diseño del adaptador de recursos no es genérico para cualquier SLEE, por lo tanto la implementación del adaptador está ligada fuertemente con el SLEE que se utiliza.

Al igual que los trabajos relacionados anteriormente descritos, el proyecto TeamCom y los trabajos desarrollados a partir de él, no especifican lineamientos que describan la mejor forma de hacer uso de las capacidad que ofrece BPEL ni de los servicios Web de carácter genérico que se necesitan para crear y componer servicios de telecomunicaciones JAIN SLEE.

RESUMEN

En este capítulo se presentó la descripción de los conceptos y tecnologías relacionados con la composición de servicios en entornos JAIN SLEE, tales como SOA, SDP y la especificación JAIN SLEE; además se realizó un análisis de los trabajos relacionados con el desarrollo, composición y despliegue de servicios convergentes, en los cuales se utilizaban diferentes plataformas y mecanismos de composición de servicios. Los trabajos que se analizan incluyen el entorno de creación de servicios StarSCE y su correspondiente entorno de ejecución StarSLEE, los cuales centran su esfuerzo en la creación de servicios de telecomunicaciones a manera de servicios Web enfocándose en cumplir los requerimientos que exigen los servicios de telecomunicaciones y que los servicios Web tradicionales no satisfacen. También se analiza el proyecto SPICE el cual se caracteriza por plantear una plataforma robusta para el desarrollo de servicios de valor agregado para entorno móviles y en donde se aborda la composición de servicios a través de un motor de composición automática (ACE). Por último, es analizado el proyecto TeamCom y los trabajos desarrollados a partir de él, en donde se plantean mecanismos de composición de servicios para entornos JAIN SLEE a través de la adaptación de las capacidades que proporciona BPEL.

Para cada uno de los trabajos analizados, se mencionan sus aportes y sus desventajas en el marco de este proyecto de grado, de las cuales se resaltaron las implementaciones propietarias, la baja eficacia sobre servicios en tiempo real, la falta de documentación de las diferentes plataformas y sobre todo la carencia de un conjunto de lineamientos que defina los pasos a seguir para realizar composición de servicios de telecomunicaciones en entornos JAIN SLEE.

Capítulo III

EVALUACIÓN DE ENTORNOS

Los entornos de desarrollo y composición de servicios para JAIN SLEE, son herramientas que permiten la creación, y manejo de componentes y servicios JAIN SLEE. Estas herramientas proveen una serie de interfaces que simplifican la creación de componentes JAIN SLEE como: Especificaciones de Perfil, Eventos, SBB, Descriptores de Servicio XML y Unidades Desplegables. Algunas de ellas tienen capacidades adicionales que permiten el despliegue y puesta a prueba de los servicios que son desarrollados por medio de las mismas, reduciendo de forma significativa el tiempo que le toma a un desarrollador la creación y el despliegue de nuevos servicios de valor agregado.

Actualmente es posible encontrar una gran variedad de entornos de desarrollo para JAIN SLEE, sin embargo gran parte de estos son de carácter propietario, por lo tanto dado que dentro del marco de este proyecto de grado es de interés el software de libre distribución, en esta sección serán presentados tres herramientas de libre distribución que permiten desarrollar componentes JAIN SLEE.

Estas herramientas se presentan en forma de extensiones o Plug-in del entorno de desarrollo integrado Java (IDE) Eclipse. Eclipse es un IDE desarrollado por IBM y se caracteriza por ser una plataforma extensible, es decir, que es posible añadir nuevas capacidades y funcionalidades que posibilitan el enriquecimiento de las aplicaciones que allí se desarrollan. Por lo tanto, los desarrolladores pueden crear sus propias aplicaciones a través de Eclipse y beneficiarse de todas la capacidades que este IDE provee.

El hecho de que las herramientas que permiten el desarrollo y composición de servicios para JAIN SLEE se presenten como extensiones del IDE Eclipse tiene como objetivo permitir que los desarrolladores de servicios puedan tener acceso a todas las capacidades y beneficios que presentan tanto las herramientas de desarrollo Java de Eclipse, como los generadores de archivos Java y documentos XML que provee el entorno de creación de servicios.

En este capítulo se procederá a identificar los diferentes entornos de desarrollo que permiten la creación y composición de servicios para JAIN SLEE, que cumplan con las características previamente descritas. Una vez identificados, se procederá a realizar un análisis teórico-práctico de cada uno de los entornos con el fin de escoger el más adecuado para llevar a cabo la composición de servicios JAIN SLEE. Para esto, es necesario establecer algunos criterios alcanzables y medibles, a través de los cuales sea posible analizar e identificar las ventajas y desventajas de cada entorno, también será necesario desarrollar un ejemplo de prueba que permita evaluar algunos de los criterios.

3.1 Entornos de Desarrollo y Composición de Servicios para JAIN SLEE

A continuación se realiza una descripción de los diferentes entornos para desarrollo y composición de servicios JAIN SLEE de libre distribución, la cual será ampliada posteriormente en la sección de evaluación.

3.1.1 EclipSLEE

EclipSLEE [46] es un entorno de creación de servicios (SCE), que permite el rápido desarrollo de servicios de valor agregado para JAIN SLEE. EclipSLEE es un Plug-in de JAIN SLEE para el entorno de desarrollo Eclipse, el cual provee una serie de interfaces que simplifican la creación de componentes de JAIN SLEE como: Especificaciones de Perfil, Eventos, SBB, Descriptores de Servicio XML y Unidades Desplegables. Con este Plug-in los desarrolladores pueden construir, desplegar y probar servicios completos de forma fácil y rápida.

EclipSLEE crea la estructura de las clases Java para eventos, SBB y perfiles. Por lo tanto, los desarrolladores deben agregar la lógica de servicio a estas estructuras. Esta herramienta también permite realizar cambios específicos a los elementos de JAIN SLEE a través de interfaces durante el proceso de desarrollo. El Plug-in genera y valida los descriptores XML para que los componentes sean correctos. Vale la pena resaltar que EclipSLEE es un subproyecto dirigido por la comunidad de la iniciativa Mobicents, la cual emite actualizaciones y trabajos relacionados con esta herramienta.

Una de las características más importantes que presenta EclipSLEE es la capacidad de realizar una autocompilación del código fuente que se desarrolla a través de la herramienta frente a cualquier cambio del mismo. Esto quiere decir que cada vez que el desarrollador realiza un cambio en la lógica del servicio que se encuentra desarrollando con la herramienta, se realiza una actualización de los diferentes archivos de despliegue en donde se encuentra encapsulada la lógica del servicio, incluyendo los cambios que se acaban de realizar. Esta característica representa una gran ventaja para los desarrolladores de servicios debido a que dentro del proceso de creación de servicios, frecuentemente es necesario llevar a cabo cambios en la lógica de los mismos para realizar pruebas de funcionamiento y sin la capacidad de autocompilación, el desarrollador se vería obligado a eliminar de forma manual los archivos de despliegue que encapsulan la lógica del servicio y finalmente volver a compilar el código fuente para generar los nuevos archivos que incluyan los cambios realizados.

Otra característica que presenta este entorno, es la posibilidad de realizar un despliegue directo de los servicios que se desarrollan a través de él en un entorno de ejecución local. Esta característica representa una reducción en el tiempo de desarrollo del servicio, dado que de otra forma sería necesario hacer uso de las interfaces de gestión del SLEE o de forzar un reinicio del servidor para poder desplegar el servicio, lo cual se convierte en un proceso tedioso para los desarrolladores. En la figura 9 se observan las interfaces de la herramienta que permiten desplegar los servicios de forma directa en un SLEE local.

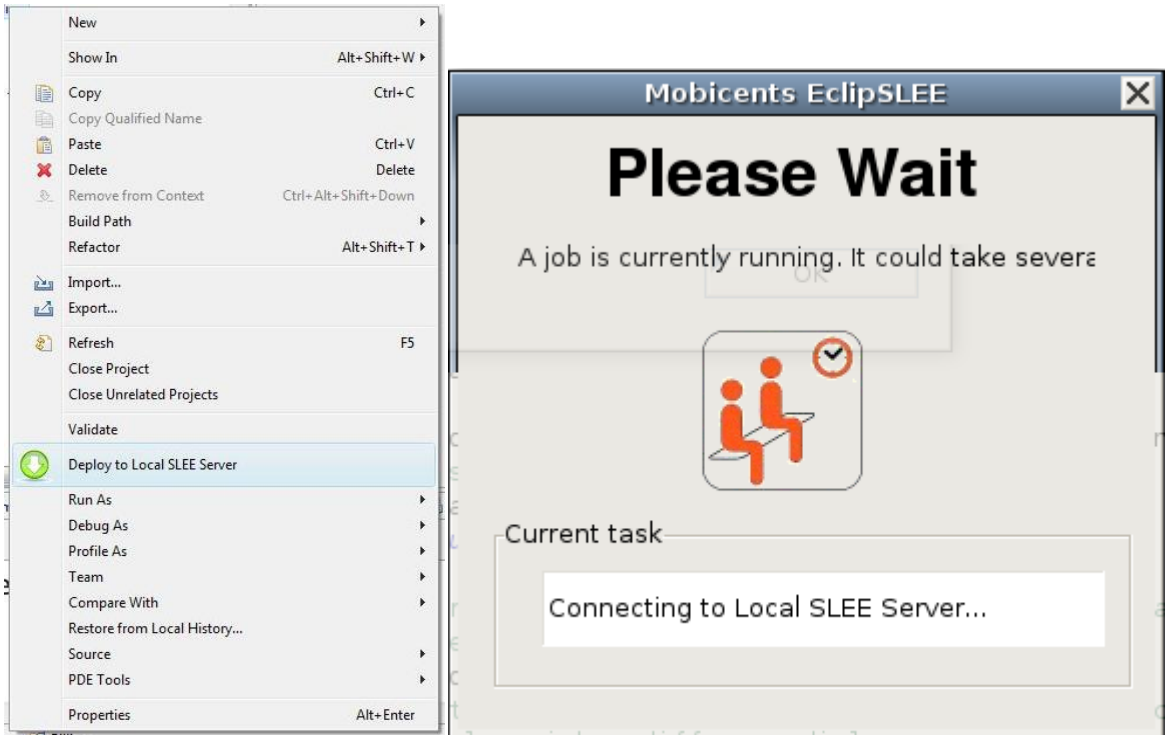

Figura 9. *Interfaces de Despliegue de Servicios de EclipSLEE* [46].

EclipSLEE proporciona también, una visualización gráfica de la estructura de los servicios que se desarrollan a través de él, donde es posible observar las relaciones que se definen entre los SBB, especificaciones de perfil, eventos y adaptadores de recursos. Esta interfaz provee la opción de exportar la imagen como un archivo PNG. La figura 10 muestra dicha interfaz gráfica.

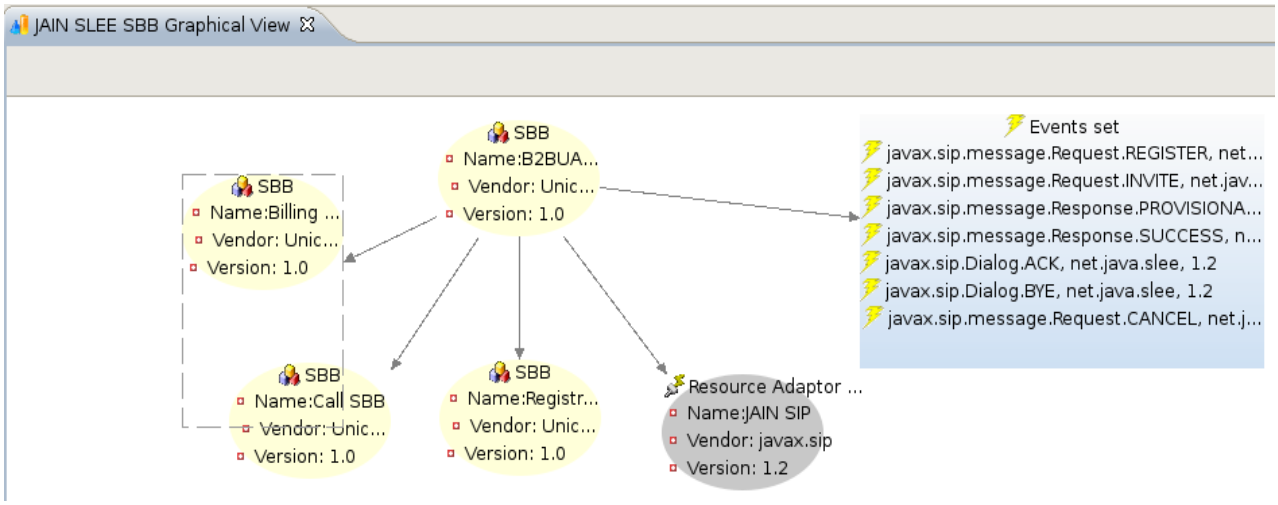

Figura 10. *Interfaz de Visualización de la Estructura del Servicio* [46].

3.1.2 Entorno de Creación de Servicios de Alcatel-Lucent (SCE-SE)

El SCE-SE de Alcatel-Lucent [47] es un entorno de desarrollo robusto que pretende la creación de servicios de telecomunicaciones y componentes JAIN SLEE como Especificaciones de Perfil, Eventos, SBB, Descriptores de Servicio XML y Unidades Desplegables. Además, el SCE-SE tiene la posibilidad de interactuar con la plataforma abierta de servicios (OSP) [48] de Alcatel-Lucent, la cual enriquece los componentes JAIN SLEE con características de administración, como agendas de programación, control de acceso, gestión de objetos, alarmas y estadísticas.

Una característica importante del SCE-SE de Alcatel-Lucent es que ofrece la opción de implementar los SBB como una máquina de estados, donde a través de una interfaz gráfica es posible generar las estructuras de código que determinan el funcionamiento del SBB y su orden lógico. En la figura 11 se observa la opción que permite crear los SBB como una máquina de estados y en la figura 12 se observa la interfaz gráfica que permite la creación de los diferentes estados del SBB.

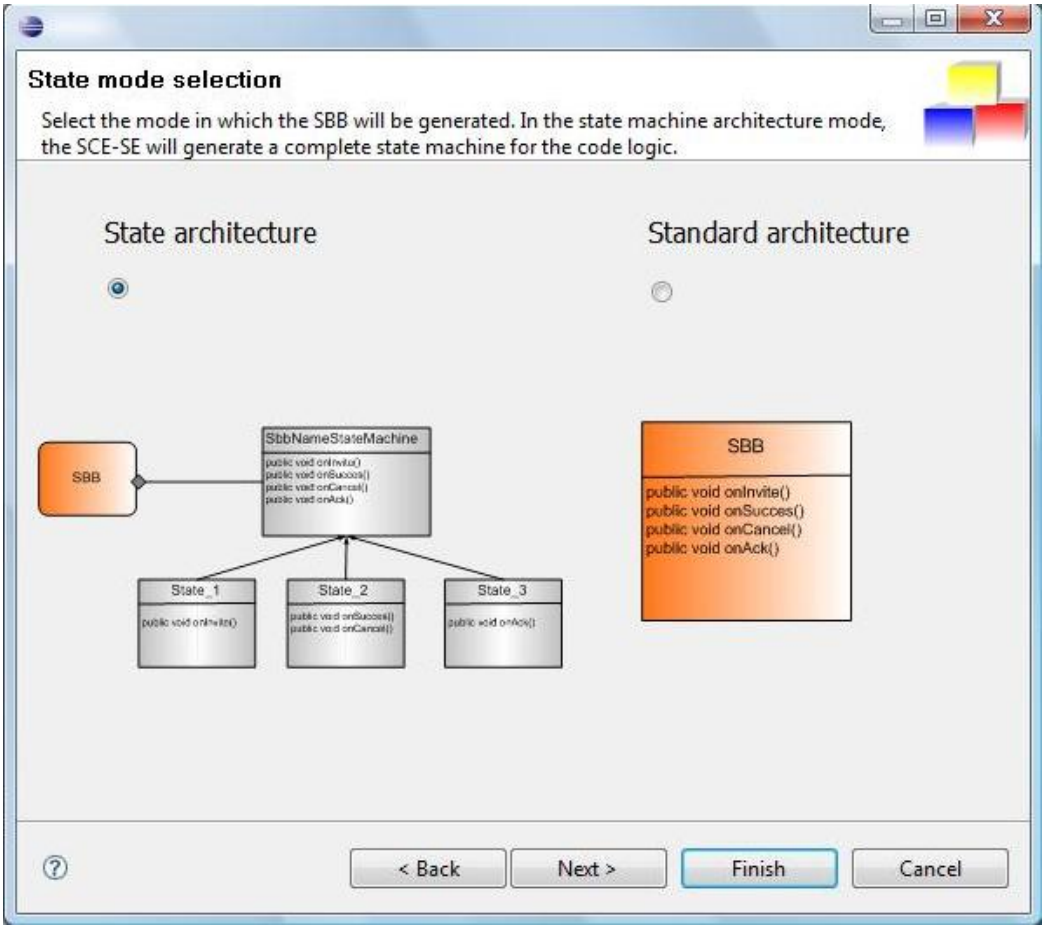

Figura 11. *Crear un SBB como una Máquina de Estados* [47].

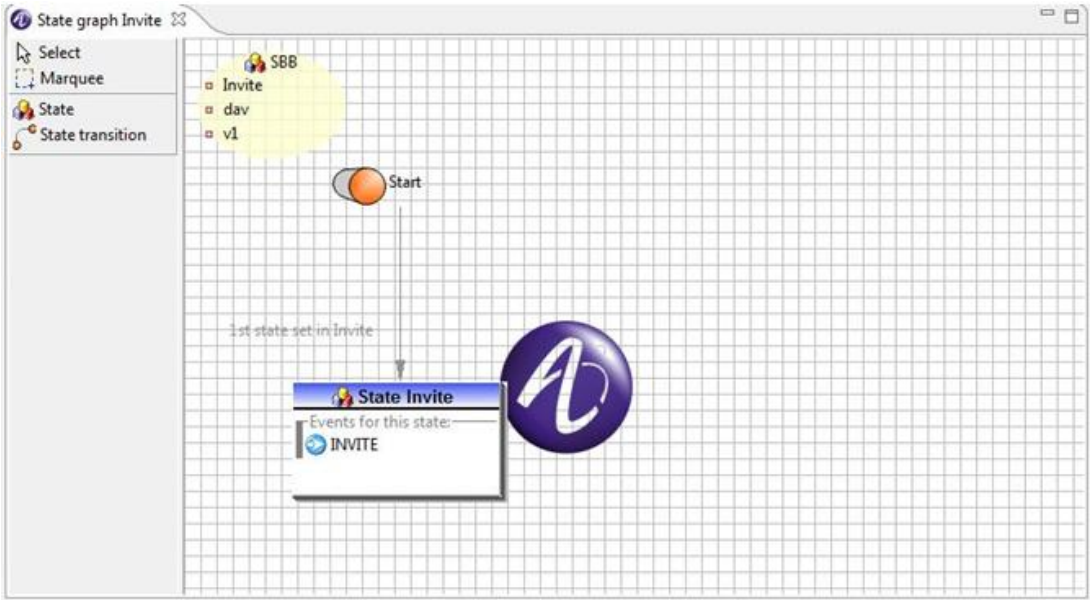

Figura 12. *Interfaz Gráfica de Estados de un SBB* [47].

Otra característica única de este entorno es que proporciona la opción de crear adaptadores de recursos y de su estructura lógica. También, genera de forma automática código fuente necesario en los SBB para utilizar las funciones comunes a todos los servicios que proporciona JAIN SLEE como temporizadores, trazas y eventos genéricos. Es de resaltar también, que este entorno permite añadir diferentes componentes JAIN SLEE previamente desarrollados a los servicios que se crean a través de él logrando evitando que todos los componentes necesarios en un servicio deban ser creados desde cero. En la figura 13 se observa dicha capacidad.

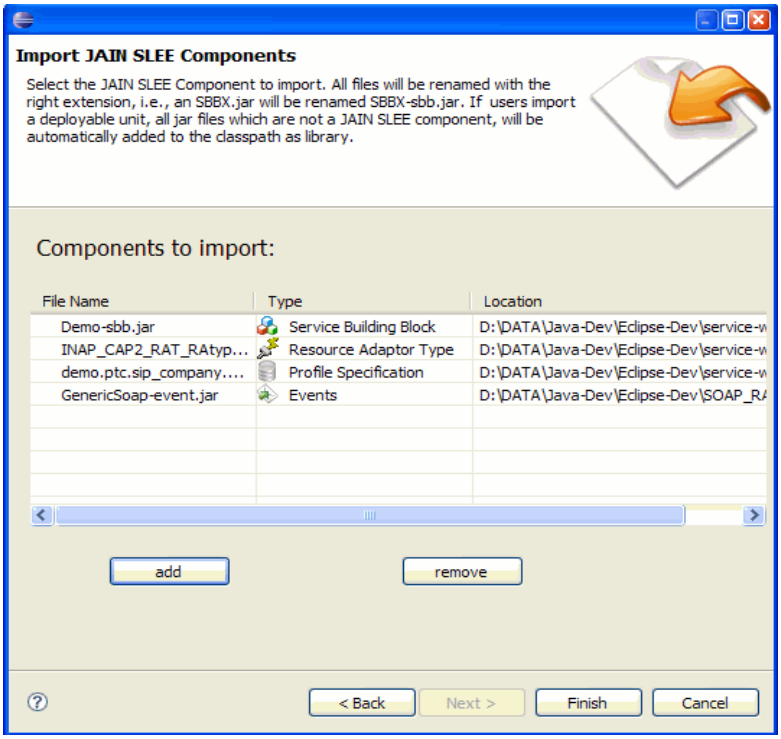

Figura 13. *Importar Componentes JAIN SLEE* [47].

Este trabajo tiene aportes importantes entre los cuales se destaca, la implementación de un entorno robusto con variedad de utilidades para creación de servicios, interfaces graficas de fácil manejo, integración con el reconocido entorno de desarrollo Eclipse, y como la principal ventaja el ser un software de libre distribución.

Como una gran desventaja de este trabajo se puede citar la carencia de una buena documentación, específicamente en la aplicación de la herramienta en proyectos que garantizan su buen uso y sus capacidades funcionales de implementación.

3.1.3 OpenCloud JAIN SLEE Plug-in

El Plug-in JAIN SLEE para el entorno de desarrollo Java Eclipse desarrollado por OpenCloud [49], está diseñado para facilitar la creación de los siguientes componentes JAIN SLEE:

- Especificaciones de Perfil
- Eventos
- Bloques Constructores de Servicio (SBB)
- Descriptores de Servicio XML
- Unidades Desplegables

Con este Plug-in los desarrolladores pueden construir servicios completos de una forma fácil y rápida. El Plug-in se encarga de asegurar que los descriptores XML sean creados de forma correcta y de crear la estructura de las clases Java en donde los desarrolladores añadirán la lógica de los servicios.

Para poder crear componentes JAIN SLEE usando el Plug-in OpenCloud, es necesario crear un proyecto, y todos los componentes externos deben hacerse disponibles para el mismo.

A diferencia de los entornos de desarrollo descritos anteriormente, el Plug-in OpenCloud no ofrece ninguna característica aparte de la capacidad de crear los componentes básicos necesarios para el desarrollo de servicios JAIN SLEE.

Es necesario resaltar la falta de ejemplos claros en los cuales se destaque la aplicación de esta herramienta en el desarrollo y composición de servicios de telecomunicaciones.

3.2 Criterios para la Evaluación de Entornos

Para realizar el análisis teórico-práctico de los diferentes entornos de desarrollo para la creación y composición de servicios JAIN SLEE, se definieron unos criterios basados en las recomendaciones dadas en el artículo [50], las cuales están relacionadas con la evaluación y análisis de herramientas software. Las características que se indican en [51] tienen en cuenta las capacidades y funcionalidades que debe tener una herramienta software a partir de su especificación, así como su comportamiento y la facilidad de manejo que debe presentar a un usuario. Con base en lo anterior, los criterios que se tienen en cuenta para el análisis y evaluación de los diferentes entornos de desarrollo para la creación y composición de servicios JAIN SLEE son los siguientes:

- **Especificación de la Herramienta:** La especificación de la herramienta es un tipo de documentación que ayuda a que los diferentes tipos de usuarios, como usuarios de negocio y usuarios técnicos, no tengan descripciones ambiguas del comportamiento y funcionalidad de la misma. También provee información acerca de la forma de como interactuar con la herramienta y de las distintas condiciones de su utilización. Las preguntas a través de las cuales se pretende utilizar este criterio para la evaluación de las herramientas son las siguientes:

    - *¿La herramienta posee un documento de especificación?*
    - *¿La especificación describe claramente el propósito de la herramienta?*
    - *¿La especificación describe la funcionalidad desde el punto de vista técnico?*

- **Autonomía:** La autonomía es la característica que describe el nivel de independencia de la herramienta de otras aplicaciones, para poder llevar a cabo su proceso de funcionamiento de forma completa. Las preguntas a través de las cuales se pretende utilizar este criterio para la evaluación de las herramientas son las siguientes:

    - *¿El funcionamiento de la herramienta depende de otra aplicación?*
    - *¿La creación de servicios a través de la herramienta es independiente de cualquier otra aplicación?*
    - *¿El despliegue y ejecución de los servicios creados por medio de la herramienta depende de otra aplicación?*

- **Definición de la Estructura del Servicio:** La definición de la estructura del servicio, es un criterio que pretende determinar el nivel de fidelidad que tiene la herramienta hacia la especificación JAIN SLEE, en cuanto a la creación y definición de los diferentes elementos que componen la estructura de un servicio JAIN SLEE. Las preguntas a través de las cuales se pretende utilizar este criterio para la evaluación de las herramientas son las siguientes:

    - *¿Tienen los servicios creados por la herramienta todos los elementos definidos en la especificación JAIN SLEE?*
    - *¿La herramienta soporta el uso de las funciones comunes a todos los servicios definidos en la especificación JAIN SLEE?*

- **Multiplataforma:** Criterio que determina la independencia de la herramienta del sistema operativo sobre el cual se está trabajando. La pregunta mediante la cual se pretende utilizar este criterio para la evaluación de de las herramientas es la siguiente:

    - *¿Es la herramienta independiente del sistema operativo sobre el cual trabaja?*

- **Soporte técnico:** Mediante este criterio se pretende evaluar la calidad del soporte técnico con que cuenta la herramienta. Inicialmente se busca determinar si la herramienta cuenta con algún medio que brinde soporte técnico al usuario, e identificar el tipo de medio de soporte, es decir un sitio Web, e-mail, teléfono, foro,

etc. Posteriormente se evalúa la calidad del servicio de soporte técnico, basándose en la efectividad de las soluciones propuestas.

- *¿La herramienta dispone de algún tipo de soporte técnico?*
- *¿Qué tan efectivo es el soporte técnico brindado?*

- **Versión estable:** Con este criterio se busca evaluar la estabilidad de la herramienta, basándose en definir si la herramienta presenta un correcto funcionamiento sin ocurrencia de fallos. Con este objetivo se incluye dentro de este criterio el establecer si la herramienta se ha utilizado en algún proyecto, y ha tenido un correcto funcionamiento que garantice su estabilidad. Finalmente dado que cada herramienta posee varias versiones, o funciona sobre alguna versión de eclipse, se debe definir cuál de ellas es la más estable.

  - *¿La herramienta funciona correctamente sin presentar fallos?*
  - *¿Existen proyectos que utilicen la herramienta y garanticen su funcionamiento estable?*
  - *¿Cuál es la versión más estable que tiene la herramienta?*

- **Instalación:** Para evaluar la instalación de las herramientas como primera medida se tiene en cuenta la existencia de alguna guía de instalación o documentación al respecto. También se debe evaluar el contenido del documento, es decir si este permite realizar una instalación satisfactoria de la herramienta o si se presentan errores o inconvenientes, los cuales se deben describir detalladamente.

  - *¿Existe documentación relacionada con la instalación de la herramienta?*
  - *¿De acuerdo a la documentación sobre la instalación se presentan inconvenientes? ¿Cuáles?*
  - *¿Se logra una correcta instalación de la herramienta?*

- **Características adicionales:** Por medio de este criterio se pretende evaluar las herramientas de acuerdo a las funciones adicionales que ofrecen, teniendo como referencia las funciones básicas definidas en la especificación JAIN SLEE. Por lo tanto se busca listar las funciones adicionales que ofrece cada herramienta y explicar claramente su función.

  - *¿La herramienta presenta características o funciones adicionales a las planteadas por la especificación JAIN SLEE?*

3.3 Servicio de Prueba de los Entornos

El proceso llevado a cabo para la evaluación de los diferentes entornos consiste en la implementación de un servicio simple a través de cada uno de ellos, en el cual se pueda evidenciar la composición de servicios. De esta forma es posible analizar el funcionamiento de cada uno de los entornos y así, poder evaluar los criterios que hacen referencia a la funcionalidad del mismo, como *Definición de la Estructura del Servicio, Versión Estable, Instalación y Características Adicionales.*

El servicio utilizado para la evaluación de los entornos, es un servicio que se compone de un servicio de Registro SIP, un servicio *Back-to-Back User Agent* (B2BUA) SIP y una Especificación de Perfil JAIN SLEE. En este servicio el usuario tiene la posibilidad de

registrarse en el SLEE a través del servicio de Registro SIP, el cual almacena en la Especificación del Perfil la información de contacto del usuario. De esta manera los usuarios tienen la posibilidad de establecer una llamada SIP con otros usuarios previamente registrados, a través del servicio B2BUA.

La figura 14 ilustra la lógica de funcionamiento del servicio de prueba, describiendo paso a paso la secuencia que se lleva a cabo entre el SLEE y los clientes SIP.

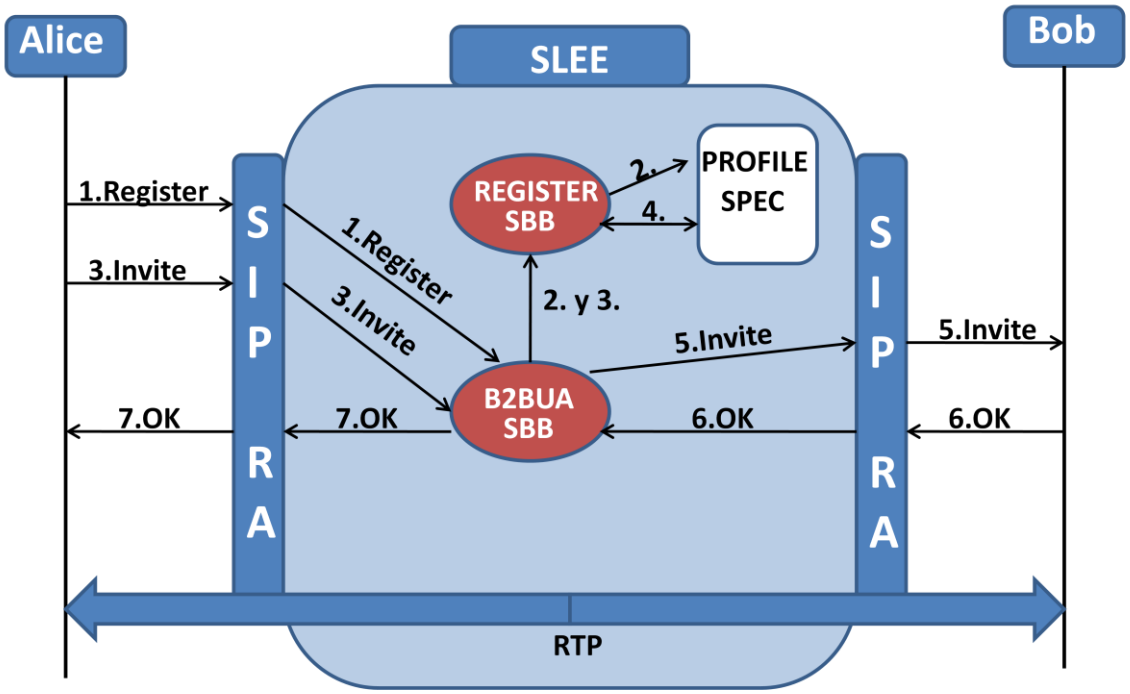

Figura 14: *Lógica del Servicio de Prueba* [15].

Inicialmente, el usuario *Alice* envía una petición de registro que es procesada por el SBB B2BUA quien transfiere la información al SBB de Registro SIP, el cual almacena la información de contacto en la Especificación de Perfil. Una vez realizado el registro, el usuario *Alice* realiza una invitación para una conversación al usuario *Bob*, el cual se encuentra previamente registrado en el SLEE, a través de una petición SIP *Invite*. La petición *Invite* es recibida por el SBB B2BUA quien a través de los métodos del SBB de Registro, se encarga de verificar que el usuario *Bob* se encuentra registrado y de obtener su información de contacto en la Especificación del Perfil. Finalmente, se redirige la petición *Invite* hacia el usuario *Bob* con la correcta información de contacto, logrando así un correcto establecimiento de la conversación.

3.4 Evaluación de Entornos de Desarrollo y Composición de Servicios JAIN SLEE

A continuación se procederá a evaluar los diferentes entornos de desarrollo y composición de servicios para JAIN SLEE, con base en los diferentes criterios de evaluación previamente definidos en este capítulo. Esta evaluación se realizará con el fin de determinar la herramienta más adecuada para llevar a cabo la composición de servicios de telecomunicaciones en entornos JAIN SLEE.

3.4.1 Evaluación de Alcatel-Lucent (SCE-SE)

- **Especificación de la Herramienta:**

  - *¿La herramienta posee un documento de especificación?*

    Esta herramienta tiene una guía de usuario muy completa [47], publicada por los desarrolladores de la herramienta, esta guía contiene descripciones de la mayoría de las funciones y componentes de la herramienta.

  - *¿La especificación describe claramente el propósito de la herramienta?*

    En el documento de especificación se plantea de forma breve pero muy clara el propósito principal de la herramienta, el cual es el desarrollo de servicios JAIN SLEE a través de la creación de los diferentes componentes definidos en la especificación.

  - *¿La especificación describe la funcionalidad desde el punto de vista técnico?*

    El documento de especificación describe detalladamente cada función de la herramienta de manera que el usuario puede seguir sus pasos para crear componentes por medio de ella, tales como eventos, especificaciones de perfil, SBB, adaptadores de recursos, descriptores de servicios y unidades desplegables. Por otra parte este documento no explica el funcionamiento interno de la herramienta tal como código fuente, ni solución de fallos.

- **Autonomía:**

  - *¿El funcionamiento de la herramienta depende de otra aplicación?*

    Dado que la forma de distribución de esta herramienta es como un Plug-in del IDE Eclipse, como consecuencia hay una total dependencia de la herramienta sobre el entorno de desarrollo Eclipse.

  - *¿La creación de servicios a través de la herramienta es independiente de cualquier otra aplicación?*

    La herramienta puede crear, manipular y gestionar componentes JAIN SLEE sin necesidad de depender de alguna otra herramienta que la asista. Esto es posible gracias a que en la lógica de la herramienta se encuentran definidos todas las interfaces que permiten crear paso a paso los diferentes componentes JAIN SLEE.

  - *¿El despliegue y ejecución de los servicios creados por medio de la herramienta depende de otra aplicación?*

    Esta herramienta permite generar servicios y componentes JAIN SLEE, además agrega algunas funciones de gestión, sin embargo no puede

ejecutar dichos servicios. Por lo tanto esta herramienta necesita de una aplicación SLEE que despliegue y ejecute los servicios desarrollados.

- **Definición de la Estructura del Servicio:**

  - *¿Tienen los servicios creados por la herramienta todos los elementos definidos en la especificación JAIN SLEE?*

    Dado que la implementación de esta herramienta está basada en la especificación 1.0 de JAIN SLEE, los componentes creados contienen todos los elementos que define la especificación tales como clases, eventos, perfiles, descriptores de servicio, SBB y unidades desplegables.

  - *¿La herramienta soporta el uso de las funciones comunes a todos los servicios definidos en la especificación JAIN SLEE?*

    En la creación de servicios la herramienta ofrece las siguientes funciones comunes a todos los servicios: alarmas, temporizadores y trazas. Estas funciones pueden ser utilizadas por los SBB y las tareas que realizan dependen de la lógica de funcionamiento implementada por el desarrollador. Alcaltel-Lucent SCE-SE proporciona de forma automática la inicialización de los parámetros necesarios para la utilización de todas las funciones.

- **Multiplataforma:**

  - *¿Es la herramienta independiente del sistema operativo sobre el cual trabaja?*

    Esta herramienta es independiente del sistema operativo, puesto que esta implementada sobre lenguaje Java, de esta manera solo necesita una maquina virtual de Java, la cual funciona sobre cualquier sistema operativo. Además debido a que funciona a manera de un Plug-in del entorno de desarrollo Eclipse, puede funcionar sobre cualquier sistema operativo ya que Eclipse proporciona distribuciones soportadas sobre los sistemas operativos más utilizados.

- **Soporte técnico:**

  - *¿La herramienta dispone de algún tipo de soporte técnico?*

    Esta herramienta cuenta con un grupo en una comunidad Web [51], la cual realiza foros acerca de problemas con la herramienta, e informa sobre nuevas actualizaciones. También se cuenta con un e-mail de la persona que desarrollo la herramienta, y por lo tanto posee un amplio conocimiento de ella.

  - *¿Qué tan efectivo es el soporte técnico brindado?*

    El medio de soporte técnico no es muy efectivo puesto que está bastante desactualizado, su último acontecimiento se presento en noviembre del

2008, y además ofrece actualizaciones en otros enlaces que no están habilitados. Respecto al contacto mencionado anteriormente, no se obtuvo respuesta alguna a la solicitud de soporte.

- **Versión estable:**

  - *¿La herramienta funciona correctamente sin presentar fallos?*

    Esta herramienta presenta los siguientes fallos:

    - ✓ Existe un problema con el editor grafico ya que cuando se quiere crear un SBB mediante la opción grafica de la máquina de estados, el editor no funciona. Sin embargo se encontró una solución que permite su utilización, la cual consiste en ejecutar el código fuente de la herramienta como un proyecto de Eclipse, esto abre otra instancia de Eclipse con todas las capacidades y los problemas del editor gráfico corregidos.

    - ✓ No se puede obtener un correcto despliegue de la interfaz de gestión de la OSP, la cual permite la gestión de parámetros del servicio cuando este se encuentra en ejecución. Esto se debe a que el entorno de ejecución de la OSP no está incluido dentro del Plug-in. Por lo tanto la solución sería ejecutar la OPS en conjunto con la herramienta, sin embargo la OSP no se encuentra disponible para su libre utilización.

    - ✓ Al compilar un servicio desarrollado con la herramienta se produce un error en la siguiente línea de código perteneciente al código fuente de un SBB:

      ```
      sbbId = (SbbID) jndiContext.lookup("slee/sbb/id")
      ```

      Esta línea de código permite obtener el identificador para el SBB. Por lo tanto se debe utilizar la variable de contexto definida de la siguiente manera:

      ```
      Context myEnv = (Context) new InitialContext().lookup("java:comp/env");
      ```

      Este objeto proporciona los métodos que permiten obtener una identificación del SBB.

  - *¿Existen proyectos que utilicen la herramienta y garanticen su funcionamiento estable?*

    Hasta el momento no se conocen proyectos que utilicen o hayan utilizado la herramienta.

  - *¿Cuál es la versión más estable que tiene la herramienta?*

Los desarrolladores de la herramienta recomiendan la versión 3.2 de Eclipse y la versión 1.4.0 de esta herramienta.

- **Instalación:**

  - *¿Existe documentación relacionada con la instalación de la herramienta?*

    Existe una guía de usuario [47] que contiene todos los pasos detallados de la instalación de la herramienta. De forma general los pasos son los siguientes:

    - Instalación de Eclipse y de los complementos necesarios.
    - Copiar los archivos necesarios en las carpetas *Plug-in* y *Features* de Eclipse.
    - Configuración de los parámetros de la OSP.

    Es de resaltar que en la guía no se explica la manera de resolver los problemas presentados por el editor gráfico de SBB descrito anteriormente.

  - *¿De acuerdo a la documentación sobre la instalación se presentan inconvenientes? ¿Cuáles?*

    Solo tiene un inconveniente al descargar el Plug-in por medio del IDE Eclipse, por lo tanto es necesario descargarlo de forma directa desde [51].

  - *¿Se logra una correcta instalación de la herramienta?*

    A pesar de los inconvenientes de descarga de la herramienta y del editor gráfico de SBB, cuyas soluciones se describen en las secciones anteriores, se obtiene una correcta instalación de la herramienta.

- **Características adicionales:**

  - *¿La herramienta presenta características o funciones adicionales a las planteadas por la especificación JAIN SLEE?*

    Esta herramienta ofrece una serie de características adicionales, las cuales se pueden realizar a través de interfaces bien definidas y que paso a paso permiten crear y gestionar los diferentes componentes definidos en la especificación JAIN SLEE además de los componentes relacionados con la OSP de Alcatel-Lucent. Las características son las siguientes:

    - ✓ Permite crear adaptadores de recursos.
    - ✓ Permite crear y agregar alarmas de gestión sobre los componentes.
    - ✓ Permite crear y agregar parámetros que proporcionan estadísticas acerca de los componentes.
    - ✓ Permite implementar SBB basándose en maquinas de estado, mediante una sencilla interfaz grafica de usuario.
    - ✓ Permite gestionar los servicios en tiempo de ejecución mediante la OSP que contiene esta herramienta.

- ✓ Cuando se crea un nuevo SBB, esta herramienta genera los atributos básicos de la especificación, como el contexto, el perfil, las actividades y las funciones de temporizadores, alarmas y trazas, que define la especificación JAIN SLEE.

3.4.2 Evaluación de EclipSLEE

- **Especificación de la Herramienta:**

  - *¿La herramienta posee un documento de especificación?*

    Esta herramienta tiene una guía de usuario Web [52] publicada por los desarrolladores de la herramienta, esta guía contiene descripciones de la mayoría de las funciones y componentes de la herramienta, tales como la creación de proyectos, eventos, especificaciones de perfil, SBB y unidades desplegables. Además, existe un video tutorial Web [53] que ilustran el funcionamiento de la herramienta.

  - *¿La especificación describe claramente el propósito de la herramienta?*

    En el sitio Web de la herramienta se plantea de forma breve pero muy clara el propósito principal de la herramienta, el cual es el desarrollo de servicios JAIN SLEE a través de la creación de los diferentes componentes definidos en la especificación.

  - *¿La especificación describe la funcionalidad desde el punto de vista técnico?*

    El documento de especificación Web describe las capacidades funcionales de la herramienta que permiten crear los componentes JAIN SLEE, tales como eventos, especificaciones de perfil, SBB, adaptadores de recursos, descriptores de servicios y unidades desplegables, de manera que el usuario puede seguir sus pasos para crear y componer servicios JAIN SLEE por medio de ella. Sin embargo, no se dispone de documentación acerca de su código fuente, ni de cursos de acción en caso de fallos.

- **Autonomía:**

  - *¿El funcionamiento de la herramienta depende de otra aplicación?*

    Dado que la forma de distribución de esta herramienta es a manera de un Plug-in del IDE Eclipse, como consecuencia hay una total dependencia de la herramienta sobre Eclipse.

  - *¿La creación de servicios a través de la herramienta es independiente de cualquier otra aplicación?*

    La herramienta puede crear, manipular y gestionar componentes JAIN SLEE sin necesidad de alguna otra herramienta que la asista.

- *¿El despliegue y ejecución de los servicios creados por medio de la herramienta depende de otra aplicación?*

  Esta herramienta permite generar servicios y componentes JAIN SLEE, sin embargo no puede ejecutar dichos servicios debido a que el Plug-in no incluye un entorno de ejecución de servicios. Por lo tanto esta herramienta necesita de una aplicación SLEE que despliegue y ejecute los servicios desarrollados.

- **Definición de la Estructura del Servicio:**

  - *¿Tienen los servicios creados por la herramienta todos los elementos definidos en la especificación JAIN SLEE?*

    Dado que la implementación de esta herramienta está basada en la especificación 1.0 de JAIN SLEE, los componentes creados contienen todos los elementos que define la especificación tales como clases, eventos, perfiles, descriptores de servicio, SBB y unidades desplegables.

  - *¿La herramienta soporta el uso de las funciones comunes a todos los servicios definidos en la especificación JAIN SLEE?*

    La herramienta soporta la utilización de todas las funciones comunes definidas en la especificación JAIN SLEE, sin embargo, no proporciona la opción de incluir dichas funciones durante la creación de un SBB. Por lo tanto las funciones requeridas por un servicio deben ser incluidas de forma manual por parte del desarrollador, definiendo las variables y parámetros necesarios para hacer uso de estas funciones, dentro de la lógica de funcionamiento del SBB.

- **Multiplataforma:**

  - *¿Es la herramienta independiente del sistema operativo sobre el cual trabaja?*

    Esta herramienta es independiente del sistema operativo, puesto que esta implementada sobre lenguaje Java, de forma que solo necesita una maquina virtual de Java la cual funciona sobre cualquier sistema operativo. Además Eclipse puede funcionar sobre cualquier sistema operativo, y solo necesita una maquina virtual de Java.

- **Soporte técnico:**

  - *¿La herramienta dispone de algún tipo de soporte técnico?*

    Esta herramienta cuenta con un grupo en una comunidad Web [54], la cual realiza foros acerca de problemas y del funcionamiento de la herramienta, e informa sobre nuevas actualizaciones.
  - *¿Qué tan efectivo es el soporte técnico brindado?*

El medio de soporte técnico es bastante efectivo ya que los miembros de la comunidad Web tratan de dar solución a los diferentes problemas planteados en el foro de forma oportuna.

- **Versión estable:**

  - *¿La herramienta funciona correctamente sin presentar fallos?*

    Esta herramienta presenta un error al tratar de importar algún componente JAIN SLEE previamente desarrollado en un proyecto, el cual impide llevar a cabo dicha tarea. Una alternativa encontrada a este inconveniente consiste en crear el componente JAIN SLEE desde cero dentro del proyecto en el cual se está trabajando y posteriormente copiar el código fuente del componente que se quiere importar al componente recién creado.

  - *¿Existen proyectos que utilicen la herramienta y garanticen su funcionamiento estable?*

    Hasta el momento no se conocen proyectos que utilicen o hayan utilizado la herramienta, solo se conocen algunos ejemplos desarrollados.

  - *¿Cuál es la versión más estable que tiene la herramienta?*

    La versión más estable de la herramienta es la versión 1.2.4, trabajando sobre la versión de Eclipse 3.3.

- **Instalación:**

  - *¿Existe documentación relacionada con la instalación de la herramienta?*

    Existe una guía de usuario [55] que contiene todos los pasos detallados de la instalación de la herramienta, los cuales, consisten en descargar el Plug-in a través de la opción de actualización de software que proporciona Eclipse. Para esto es necesario contar con la URL donde se encuentra el Plug-in la cual es proporcionada en [55].

  - *¿De acuerdo a la documentación sobre la instalación se presentan inconvenientes? ¿Cuáles?*

    En el proceso de instalación, el único inconveniente que se presenta tiene que ver con las versiones tanto de EclipSLEE como de Eclipse, ya que si se sigue el proceso de instalación con versiones diferentes a la 1.2.4 de EclipSLEE y la versión 3.3 de Eclipse, no es posible agregar componentes externos a los proyectos creados como los adaptadores de recursos. Esto se debe a que la opción que permite realizar esta tarea no queda disponible.

  - *¿Se logra una correcta instalación de la herramienta?*

    Se obtiene una correcta instalación de la herramienta siempre y cuando se utilicen las versiones 1.2.4 de EclipSLEE y 3.3 de Eclipse.

- **Características adicionales:**

  - *¿La herramienta presenta características o funciones adicionales a las planteadas por la especificación JAIN SLEE?*

    Esta herramienta ofrece una serie de características adicionales, las cuales se pueden realizar a través de interfaces bien definidas y que paso a paso permiten crear y gestionar los diferentes componentes definidos en la especificación JAIN SLEE. Estas características son explicadas en detalle en la sección 3.1.1. Las características son las siguientes:

    - ✓ Permite visualizar la estructura de los servicios y las relaciones definidas entre los diferentes componentes JAIN SLEE.

    - ✓ Permite desplegar los servicios directamente en un entorno SLEE que se encuentre activo.

    - ✓ Realiza una autocompilación del código fuente de los servicios, lo cual evita que sea necesario borrar de forma manual los archivos creados en compilaciones anteriores.

3.4.3 Evaluación de OpenCloud JAIN SLEE Plug-in

- **Especificación de la Herramienta:**

  - *¿La herramienta posee un documento de especificación?*

    Esta herramienta tiene una guía de usuario que se encuentra dentro de la instalación de la misma. Esta guía de usuario es accesible a través de la ayuda de Eclipse y contiene descripciones de la mayoría de las funciones y componentes de la herramienta, tales como la creación de proyectos, eventos, especificaciones de perfil, SBB y unidades desplegables.

  - *¿La especificación describe claramente el propósito de la herramienta?*

    En el documento se describe de forma clara el propósito de la herramienta, el cual es el desarrollo de servicios JAIN SLEE a través de la creación de los diferentes componentes definidos en la especificación.

  - *¿La especificación describe la funcionalidad desde el punto de vista técnico?*

    El documento de especificación describe la funcionalidad de cada componente de la herramienta, de manera que el usuario puede seguir sus pasos para crear y componer servicios JAIN SLEE por medio de ella. Sin embargo, no se dispone de documentación acerca de su código fuente, ni de cursos de acción en caso de fallos.

- **Autonomía:**

  - *¿El funcionamiento de la herramienta depende de otra aplicación?*

Dado que la forma de distribución de esta herramienta es a manera de un Plug-in del IDE Eclipse, como consecuencia hay una total dependencia de la herramienta sobre Eclipse.

- *¿La creación de servicios a través de la herramienta es independiente de cualquier otra aplicación?*

  La herramienta puede crear, manipular y gestionar componentes JAIN SLEE sin necesidad de alguna otra herramienta que la asista.

- *¿El despliegue y ejecución de los servicios creados por medio de la herramienta depende de otra aplicación?*

  Esta herramienta permite generar servicios y componentes JAIN SLEE, sin embargo no puede ejecutar dichos servicios debido a que el Plug-in no incluye un entorno de ejecución de servicios. Por lo tanto esta herramienta necesita de una aplicación SLEE que despliegue y ejecute los servicios desarrollados.

- **Definición de la Estructura del Servicio:**

  - *¿Tienen los servicios creados por la herramienta todos los elementos definidos en la especificación JAIN SLEE?*

    Dado que la implementación de esta herramienta está basada en la especificación 1.0 de JAIN SLEE, los componentes creados contienen todos los elementos que define la especificación tales como clases, eventos, perfiles, descriptores de despliegue, etc.

  - *¿La herramienta soporta el uso de las funciones comunes a todos los servicios definidos en la especificación JAIN SLEE?*

    La herramienta soporta la utilización de todas las funciones comunes definidas en la especificación JAIN SLEE, sin embargo, no posee interfaces que permitan incluir dichas funciones durante la creación de un servicio. Por lo tanto las funciones requeridas por un servicio deben ser incluidas de forma manual por parte del desarrollador.

- **Multiplataforma:**

  - *¿Es la herramienta independiente del sistema operativo sobre el cual trabaja?*

    Esta herramienta es independiente del sistema operativo, puesto que esta implementada sobre lenguaje Java, de forma que solo necesita una maquina virtual de Java la cual funciona sobre cualquier sistema operativo. Además Eclipse puede funcionar sobre cualquier sistema operativo, y solo necesita una maquina virtual de Java.

- **Soporte técnico:**

  - *¿La herramienta dispone de algún tipo de soporte técnico?*

    Esta herramienta cuenta con un grupo en una comunidad Web [56], la cual realiza foros acerca de problemas y del funcionamiento de la herramienta, e informa sobre nuevas actualizaciones.

  - *¿Qué tan efectivo es el soporte técnico brindado?*

    El medio de soporte técnico es bastante efectivo ya que los miembros de la comunidad Web tratan de dar solución a los diferentes problemas planteados en el foro de forma oportuna.

- **Versión estable:**

  - *¿La herramienta funciona correctamente sin presentar fallos?*

    Esta herramienta presenta un error al tratar de importar algún componente JAIN SLEE previamente desarrollado en un proyecto, el cual impide llevar a cabo dicha tarea. Una alternativa encontrada a este inconveniente consiste en crear el componente JAIN SLEE desde cero dentro del proyecto en el cual se está trabajando y posteriormente copiar el código fuente del componente que se quiere importar al componente recién creado.

  - *¿Existen proyectos que utilicen la herramienta y garanticen su funcionamiento estable?*

    Hasta el momento no se conocen proyectos que utilicen o hayan utilizado la herramienta, solo se conocen algunos ejemplos desarrollados.

  - *¿Cuál es la versión más estable que tiene la herramienta?*

    La versión más estable de la herramienta es la versión 1.0.5, trabajando sobre la versión de Eclipse 3.3.

- **Instalación:**

  - *¿Existe documentación relacionada con la instalación de la herramienta?*

    Existe una guía de usuario Web [57] que contiene todos los pasos detallados de la instalación de la herramienta, los cuales consisten en descargar los archivos correspondientes al Plug-in y posteriormente copiarlos en las carpetas *Plug-in* y *Features* de Eclipse.

  - *¿De acuerdo a la documentación sobre la instalación se presentan inconvenientes? ¿Cuáles?*

    En el proceso de instalación, el único inconveniente que se presenta tiene que ver con la versión de Eclipse, ya que si se sigue el proceso de instalación con una versión diferente a la 3.3 de Eclipse, no es posible

agregar componentes externos a los proyectos creados como los adaptadores de recursos. Esto se debe a que la opción que permite realizar esta tarea no queda disponible.

- *¿Se logra una correcta instalación de la herramienta?*

  Se obtiene una correcta instalación de la herramienta.

- **Características adicionales:**

  - *¿La herramienta presenta características o funciones adicionales a las planteadas por la especificación JAIN SLEE?*

    Como se explicó en la sección 3.1.3, esta herramienta no ofrece ninguna característica adicional aparte de las funciones básicas necesarias para crear servicios JAIN SLEE.

3.5 Tabla Comparativa de Entornos de Desarrollo y Composición de Servicios JAIN SLEE

La siguiente tabla resume la evaluación realizada a cada una de las herramientas de desarrollo y composición de servicios JAIN SLEE en la sección 3.1. Adicionalmente se define una ponderación y una escala de calificación (de 1 a 5, siendo 5 el mejor resultado) sobre el comportamiento de las herramientas respecto de cada uno de los criterios tenidos en cuenta en la sección 3.2. De esta manera es posible llevar a cabo una comparación de las herramientas con la finalidad de escoger la más adecuada para realizar composición de servicios de telecomunicaciones en entornos JAIN SLEE.

Criterio y Porcentaje		**Herramienta**		
		Alcatel-Lucent (SCE-SE)	**EclipSLEE**	**OpenCloud JAIN SLEE Plug-in**
Especificación de la Herramienta	10%	5	4	4
Autonomía	5%	3	3	3
Definición de la Estructura del Servicio	15%	5	3	3
Multiplataforma	5%	5	5	5
Soporte técnico	5%	1	5	5
Versión estable	20%	3	3	3
Instalación	10%	3	3	3
Características adicionales	30%	4	4	1
Total		**3.8**	**3.6**	**2.4**

Tabla 1. *Evaluación de Herramientas de Desarrollo y Composición de Servicios JAIN SLEE*

Teniendo en cuenta la evaluación de los entornos de la sección 3.3 y la anterior tabla de resultados, a continuación se analizan las diferencias más relevantes de los resultados. El

criterio 3 obtiene un 5 para Alcatel ya que es el único que soporta alarmas, trazas y eventos estadísticos que define la especificación. El criterio 5 es calificado con un 1 para Alcatel dado que el único soporte técnico que tiene esta inactivo desde hace 1 año. El criterio 7 obtiene un 3 para todas las herramientas ya que presentan dificultades en la instalación, como por ejemplo por parte de Alcatel existen inconvenientes adicionales tales como la necesidad de ejecutar la herramienta como un proyecto del IDE Eclipse para poder acceder a la interfaz grafica de la máquina de estados de los SBB, y el no funcionamiento de la plataforma OSP; para EclipSLEE y OpenCloud existen dificultades de funcionamiento con las versiones del Plug-in y Eclipse, además el directorio de trabajo de Eclipse debe estar vacio o de lo contrario genera problemas al importar componentes JAIN SLEE . Por último, el criterio 8 es calificado con un 1 para Open Cloud ya que esta herramienta no ofrece ninguna característica adicional que complemente o amplíe las facilidades para la composición de servicios.

3.6 Resultados de Evaluación

En esta sección se analizaran las conclusiones alcanzadas después de haber realizado el proceso de evaluación de los entornos de desarrollo y composición de servicios JAIN SLEE, utilizando los diferentes criterios previamente definidos en este capítulo.

Después de realizar la evaluación de las herramientas mediante los puntajes establecidos en la tabla 1, se concluye que tanto Alcatel como EclipSLEE obtienen un puntaje superior al 70%, cualquiera de las dos herramientas cumple con los requisitos mínimos para componer servicios JAIN SLEE. Por lo tanto para seleccionar una de ellas se utiliza el criterio 8 de características adicionales, ya que este criterio marca la diferencia entre las dos herramientas.

Con base en lo anterior, a continuación se mencionan cuales son las características únicas de Alcatel y EclipSLEE, las cuales facilitan el desarrollo y composición de servicios JAIN SLEE.

- EclipSLEE:
  - Permite desplegar los servicios directamente en un entorno SLEE que se encuentre activo.
  - Realiza auto compilación del código fuente de los servicios, lo cual evita que sea necesario borrar de forma manual los archivos creados en compilaciones anteriores.
- Alcatel-Lucent (SCE-SE):
  - Permite implementar SBB basándose en maquinas de estado, mediante una sencilla interfaz grafica de usuario.
  - Cuando se crea un nuevo SBB, esta herramienta genera los atributos básicos de la especificación, como el contexto, el perfil, las actividades y las funciones de temporizadores, alarmas y trazas, que define la especificación JAIN SLEE.
  - Ofrece la opción de crear nuevos adaptadores de recursos.

- Permite importar de una forma sencilla componentes externos al proyectos, tales como adaptadores de recursos, SBB, eventos o unidades desplegables.

Las características únicas que presenta la herramienta EclipSLEE son de gran ayuda para la creación de los servicios ya que permite realizar un despliegue directo de los servicios desde el IDE Eclipse al servidor SLEE, además realiza auto compilación frente a cualquier cambio del código fuente del servicio, lo cual evita el borrado manual de los archivos generados en compilaciones anteriores. En cuanto a las características únicas de la herramienta Alcatel-Lucent (SCE-SE), se pude decir que son de gran utilidad para la composición de servicios y la organización del curso de eventos al interior de un servicio, también permiten la utilización de forma automática de las funciones comunes que provee JAIN SLEE.

Teniendo en cuenta lo anterior, se puede observar que las características únicas que presentan cada una de las dos herramientas son ventajas que pueden ser tomadas como complementarias entre sí, es decir, que tanto las ventajas de EclipSLEE, como las ventajas de Alcatel-Lucent (SCE-SE), aplicadas en conjunto facilitan en gran medida la creación y composición de servicios JAIN SLEE. Por lo tanto se concluye que la mejor forma para llevar a cabo la composición de servicios JAIN SLEE es utilizando las herramientas EclipSLEE y Alcatel-Lucent (SCE-SE) de forma conjunta. Esto es posible dado que las dos herramientas se presentan a manera de Plug-in del IDE Eclipse, por lo tanto se puede utilizar simultáneamente las capacidades de ambas herramientas a través de este IDE.

RESUMEN

En este capítulo se presentó una introducción conceptual acerca de los entornos de desarrollo y composición de servicios JAIN SLEE, describiendo tres herramientas llamadas EclipSLEE, OpenCloud JAIN SLEE Plug-in y Alcatel-Lucent SCE-SE; posteriormente fueron definidos una serie de criterios que permitieron realizar una evaluación de dichos entornos, además se definió un servicio de prueba para evaluar algunos de los criterios; esto con el fin de determinar el entorno más adecuado para llevar a cabo la composición de servicios JAIN SLEE. Sin embargo, al realizar dicha evaluación, se llego a la conclusión de que la mejor forma para componer servicios JAIN SLEE es utilizar las herramientas EclipSLEE y Alcatel-Lucent (SCE-SE) de forma conjunta.

Capítulo IV

LINEAMIENTOS PARA COMPOSICIÓN DE SERVICIOS DE TELECOMUNICACIONES EN ENTORNOS JAIN SLEE

4.1 Introducción

En este capítulo se procederá a plantear una serie de lineamientos para llevar a cabo la composición de servicios de telecomunicaciones en entornos JAIN SLEE, es decir, se propondrán una serie de pautas o directivas con el propósito de brindar a un desarrollador de servicios de telecomunicaciones JAIN SLEE una manera sencilla y organizada de componer servicios de telecomunicaciones a partir de la reutilización de servicios previamente desarrollados. Para esto, se utilizará como marco de referencia los conceptos que proporciona la Arquitectura Orientada a Servicios (SOA) para el desarrollo y composición de servicios.

La utilización de SOA como marco de referencia para el planteamiento de los lineamientos de composición se debe a los diferentes beneficios que ofrece SOA para el desarrollo de servicios. Dentro de estos beneficios se destacan los siguientes:

- Desarrollo de servicios a partir de la reutilización de servicios ya existentes.
- Interoperabilidad debido a la utilización de estándares para el desarrollo de los servicios.
- Servicios flexibles que permiten agregar o quitar componentes para generar nuevas funcionalidades.
- Agilidad y eficiencia en la definición y planteamiento de nuevos servicios a través de metodologías estandarizadas como UML [58] y RUP [59].

Dado que estos lineamientos están orientados hacia la composición de servicios JAIN SLEE, es necesario profundizar en algunos conceptos fundamentales de la especificación JAIN SLEE, tales como los componentes de los servicios y la manera en la cual se relacionan. Esto con el fin de describir de una mejor forma el proceso que se debe llevar a cabo para componer servicios de telecomunicaciones JAIN SLEE.

4.2 Lineamientos para Composición de Servicios de Telecomunicaciones en Entornos JAIN SLEE

En esta sección se definen los lineamientos para composición de servicios de telecomunicaciones en entornos JAIN SLEE. La definición de estos lineamientos se encuentra estructurada de forma que se realiza inicialmente un análisis del modelo del negocio con la finalidad de obtener los requerimientos del mismo. A partir de estos requerimientos se procede a identificar y especificar los diferentes servicios que serán compuestos en un servicio más complejo que cumpla con los requerimientos del negocio. Este proceso se encuentra basado en el modelo para la creación de servicios propuesto

por SOA en [60], además, se utiliza la especificación JAIN SLEE como referencia para definir la composición de servicios de telecomunicaciones.

4.2.1 Modelado del Negocio

Los servicios de información y de comunicaciones proveen un soporte fundamental en la construcción de soluciones tecnológicas para las empresas, tanto en el mejoramiento de procesos, como en el desarrollo de nuevas funcionalidades y servicios. El uso de SOA como marco de referencia para la creación de soluciones permite identificar y analizar los problemas y necesidades de las empresas, con la finalidad de determinar los requerimientos específicos necesarios para la construcción de una solución que satisfaga las necesidades que presentan las empresas.

Antes de la implementación de una solución, es necesario identificar cual es el problema y específicamente cuales son los requerimientos que deben ser cumplidos para crear dicha solución. SOA utiliza los conceptos que plantea RUP dentro de su proceso de modelado del negocio, con el fin de identificar los requerimientos específicos para la creación de una solución.

4.2.1.1 Pasos Típicos en el Modelado del Negocio SOA

Para llevar a cabo el proceso de modelado del negocio, SOA plantea una serie de pasos y tareas basados en los conceptos de RUP, cuya finalidad es la identificación y determinación de los requisitos específicos necesarios para la creación de una solución que cumpla con los objetivos del negocio. La secuencia típica de pasos para la creación del modelo de negocio es la siguiente:

- Análisis y entendimiento general del negocio.
- Definición de la visión del negocio.
- Creación del modelo de casos de uso del negocio.
- Definición de la arquitectura del negocio.
- Captura e identificación de requisitos y objetivos del negocio.

A continuación se realizará una descripción general de los diferentes conceptos que son utilizados en el proceso de modelado del negocio propuesto por SOA.

4.2.1.2 Visión del Negocio

La visión del negocio captura los objetivos principales o de alto nivel dentro del proceso de modelado del negocio. Esta tarea es realizada desde el punto de vista de los administradores, clientes y desarrolladores de servicios que participan en la idea de negocio, buscando un entendimiento del contexto y de las relaciones que existen en él, con el fin de identificar sus objetivos principales.

La visión del negocio se encuentra íntimamente relacionada con la construcción del modelo de casos de uso del negocio ya que identifica los propósitos fundamentales de la idea de negocio y permite tener un entendimiento del mismo desde el punto de vista de los diferentes actores que en él participan.

4.2.1.3 Casos de Uso del Negocio

El punto inicial en el proceso de desarrollo de una idea de negocio es el análisis del entorno en el cual opera el negocio. El modelo de casos de uso del negocio es una técnica usada para describir el negocio desde un punto de vista externo. De manera más formal, un caso de uso del negocio es *"…una secuencia de acciones que un negocio realiza y que produce un resultado observable para un actor del negocio particular, o que muestra como el negocio responde frente a un evento específico y que genera un beneficio para el negocio"* [59].

Para tener un mejor entendimiento de esta definición se puede explicar de la siguiente manera:

- *"…una secuencia de acciones…"* quiere decir que existe un dialogo entre el negocio y uno o varios actores externos, el cual tiene un orden o secuencia y no es de tipo unidireccional.

- *"…produce un resultado observable para un actor del negocio particular…"* quiere decir que debe haber un resultado tangible que se genera a partir del proceso de negocio y que beneficia a un actor que interactúa con el negocio desde el exterior.

- *"…muestra como el negocio responde frente a un evento específico y que genera un beneficio para el negocio…"* quiere decir que como una alternativa, el proceso de negocio puede generar un beneficio para sí mismo.

El caso de uso del negocio genera una visión completa del proceso de negocio. En la práctica, esta técnica es una manera fácil y rápida de obtener un entendimiento de alto nivel del negocio.

4.2.1.4 Arquitectura del Negocio

La arquitectura del negocio provee una visión de las partes relevantes del negocio en términos de sus productos, servicios y procesos. Es usada para identificar los componentes claves del negocio y genera una pauta para la construcción de aplicaciones, servicios y otros elementos técnicos que componen al negocio.

La arquitectura del negocio permite también, tener un entendimiento de la estructura del negocio y de la forma como sus componentes interactúan entre sí. De esta manera, es posible identificar requerimientos fundamentales de carácter técnico que son necesarios para realizar el proceso de negocio de forma satisfactoria y que cumpla con los objetivos del mismo.

4.2.1.5 Identificación de Objetivos y Requisitos del Negocio

El proceso de modelado del negocio permite la identificación y captura de los requerimientos y objetivos de una idea de negocio. Tomando como referencia el proceso de modelado del negocio propuesto por SOA en [60] se llevan a cabo una serie de pasos y tareas basadas en los conceptos planteados por RUP, que brindan un entendimiento completo de una idea de negocio y finalmente permiten identificar sus objetivos y requerimientos.

A través de los conceptos descritos anteriormente es posible capturar e identificar dichos requerimientos y objetivos. La visión del negocio brinda un entendimiento general de la idea de negocio, permitiendo determinar sus objetivos principales y de esta manera tener un punto de partida en el proceso de construcción de la solución. Los casos de uso del negocio describen los diferentes procesos que tienen lugar en el interior del negocio e identifican los diferentes actores que interactúan con el mismo, permitiendo así, conocer las condiciones y requisitos necesarios para que dichos procesos puedan ser llevados a cabo. Finalmente la arquitectura del negocio provee una visión completa de la estructura del negocio y de las relaciones e interacciones que sostienen sus componentes, permitiendo que puedan ser identificados requerimientos de carácter técnico, fundamentales en el proceso de negocio.

De esta forma es posible realizar un análisis de una idea de negocio donde se identifiquen y capturen sus objetivos y requerimientos, posibilitando así, la construcción de una solución que cumpla con dichos objetivos de forma satisfactoria.

4.2.2 Identificación de Bloques Constructores del Servicio

Dentro de la metodología de SOA, este proceso pretende definir los bloques necesarios para proveer la solución requerida [61]. Con el mismo objetivo y teniendo en cuenta que este proyecto de grado está enfocado en el desarrollo y composición de servicios en entornos JAIN SLEE, la identificación de bloques constructores del servicio es definida como una actividad que tiene como objetivo principal, definir el conjunto de SBB (Service Building Blocks) que se utilizaran para proveer la funcionalidad y requerimientos establecidos, además en esta actividad también se describe brevemente la función de cada SBB y la forma en la cual interactúan entre sí.

Para obtener una exitosa identificación de SBB se proponen una serie de pasos basados en los resultados que se obtienen al realizar las actividades de modelado del negocio descritas en la sección 4.2.1. Los pasos propuestos son los siguientes:

- Análisis de recursos existentes.
- Definición de sub-procesos.
- Definición y clasificación de SBB.
- Diagramas de secuencia.

4.2.2.1 Análisis de Recursos Existentes

Dado que dentro de una solución SOA se define como uno de los principios básicos la reutilización de componentes, en la identificación de servicios se propone como primer paso realizar un análisis de los recursos existentes, con el fin de encontrar componentes que ya hayan sido implementados por el mismo desarrollador o por terceros, y que se puedan reutilizar para el desarrollo del nuevo servicio. Dichos componentes buscados se refieren específicamente a SBB, adaptadores de recursos y unidades desplegables.

El análisis de recursos existentes debe ser realizado teniendo en cuenta los requerimientos establecidos en el literal 4.2.1.5, de manera que la búsqueda esté enfocada en encontrar componentes que se relacionen con los requerimientos, y que puedan proveer algún tipo de funcionalidad requerida.

Finalmente este análisis debe entregar, en caso de que exista, un conjunto de SBB,

adaptadores de recursos y unidades desplegables, que posteriormente será tomado en cuenta.

4.2.2.2 Definición de Sub-Procesos

Para facilitar la identificación de servicios se realiza la definición de sub-procesos, este paso consiste en tomar la secuencia de cada caso de uso del negocio definido en el literal 4.2.1.3, y descomponerlo en sub-procesos o procesos más pequeños, de manera que sea más fácil identificar los SBB que se necesitan para implementar cada caso de uso.

Es importante que la división en sub-procesos se realice de forma que se obtenga una colección de sub-procesos con granularidad fina, es decir que cada sub-proceso sea una tarea suficientemente simple y especifica.

4.2.2.3 Definición y Clasificación de los SBB

La definición de los SBB es el paso más complejo dentro de la identificación de servicios, ya que en este paso se deben definir cuáles serán los SBB que se utilizaran para implementar la solución requerida. Por lo tanto para facilitar este paso se toma como punto de partida la definición de sub-procesos realizada en el literal anterior, y luego analizando cada sub-proceso de un caso de uso del negocio, el diseñador de la solución debe lograr identificar los posibles SBB que podrían implementar la funcionalidad del caso de uso del negocio.

Con el objetivo de realizar la identificación de una manera más organizada, en este literal se plantea una clasificación de SBB, además, posteriormente son propuestas algunas recomendaciones que el desarrollador del servicio deberá tener en cuenta para una exitosa identificación de SBB.

Clasificación de SBB:

Los SBB son bloques en donde se puede implementar cualquier tipo de servicio, y según la especificación JAIN SLEE existe un solo tipo de SBB. Sin embargo en el marco de este proyecto hemos establecido una clasificación basada en el papel que juega cada SBB dentro de la composición de algún servicio. Por lo tanto los SBB se clasifican en 2 tipos:

- ***SBB atómico:*** Este SBB se caracteriza por no contener o utilizar algún otro SBB dentro de su funcionamiento. Es utilizado para una función específica que puede estar relacionada con un adaptador de recursos, además su funcionamiento debe ser lo más genérico posible de manera que cualquier otro SBB pueda acceder a su funcionamiento mediante la composición.

- ***SBB compuesto:*** El SBB compuesto tienen como característica principal el estar compuesto por SBB más pequeños comúnmente llamados SBB hijos. Dentro de su lógica de funcionamiento se encuentran definidas las relaciones que le permiten utilizar a los SBB hijos, es decir que en este tipo de SBB se lleva a cabo el proceso de composición.

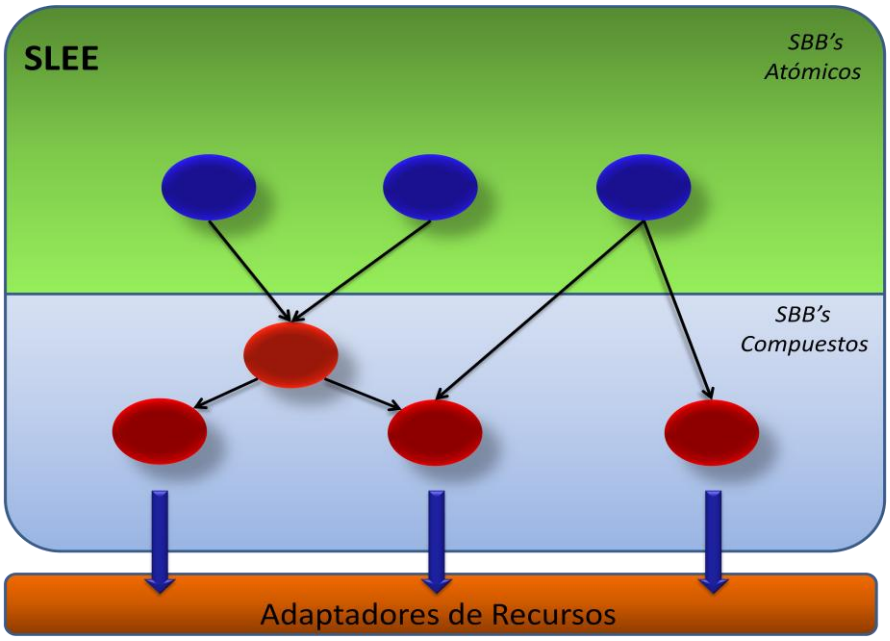

Figura 15: *Clasificación de SBB (Propia).*

En la figura 15 se observa un ejemplo en donde se relacionan diversos SBB y es posible diferenciar los SBB atómicos de los compuestos, debido a que como se explicó anteriormente, los SBB compuestos referencian y definen relaciones con otros SBB dentro de su lógica de funcionamiento, mientras que los SBB atómicos no dependen de ningún otro SBB para su funcionamiento.

Recomendaciones para clasificar SBB:

Dado que la identificación de SBB es un paso fundamental dentro del desarrollo del servicio, a continuación se darán algunas recomendaciones que el desarrollador deberá tener en cuenta para obtener fácilmente una buena identificación de SBB:

- Recordar que un SBB es una componente software (archivos .java y .xml), que ejecuta una lógica basada en los eventos que recibe y envía. Sin embargo un SBB también puede implementa lógica sin estar necesariamente ligado a eventos.
- Antes de identificar los SBB debemos revisar detenidamente el análisis de recursos existentes realizado en el literal 4.2.2.2, puesto que es posible que algún sub-proceso o funcionalidad requerida ya este implementada en algún SBB, lo cual agiliza en gran medida la etapa de implementación del servicio.
- Es recomendable que primero se identifiquen los SBB atómicos dentro de los sub-procesos del caso de uso del negocio, y posteriormente sean identificados los SBB compuestos que harán uso de los SBB atómicos mediante métodos de composición de servicios.
- Se recomienda si es posible que cada proceso que requiera comunicación con un adaptador de recursos específico sea implementada por un solo SBB atómico. De esta manera evitamos una tediosa mezcla de protocolos dentro de la implementación del servicio.
- No es necesario que en cada sub-proceso del caso de uso del negocio se identifique un SBB, ya que se puede presentar el caso de que diferentes sub-

proceso hagan referencia al mismo SBB.

- Cada SBB atómico que se identifique debe ser lo más simple posible respecto a su funcionalidad, además debe ser lo más genérico posible para así contribuir a diferentes servicios y desarrolladores.
- Cada SBB atómico debe ser independiente de los otros SBB que se identifiquen en este paso, pueden existir relaciones entre ellos pero se recomienda que no exista una fuerte dependencia entre su funcionamiento.
- Una vez se identifiquen los SBB atómicos con base en los sub-procesos de un caso de uso del negocio, posteriormente los sub-procesos que corresponden a la organización lógica de los SBB atómicos, serán implementados finalmente dentro de un SBB compuesto que corresponde al servicio completo.

4.2.2.4 Diagramas de Secuencia

Luego de identificar los SBB que serán utilizados para desarrollar el servicio requerido, es necesario realizar los diagramas de secuencia de cada caso de uso del negocio teniendo en cuenta los SBB identificados anteriormente.

Para construir un diagrama de secuencia se emplean los SBB definidos para cada caso del negocio, luego se dibujan las relaciones entre ellos las cuales son llamados de métodos o eventos de un SBB a otro, siguiendo la secuencia del flujo del servicio. Estos diagramas de secuencia se realizan basándose en la notación de UML.

4.2.3 Especificación de SBB

La especificación de SBB tiene como objetivo realizar un análisis más detallado de los SBB identificados en la sección anterior, es decir que en este paso el desarrollador debe tomar cada SBB identificado y describir las características más importantes de este, teniendo en cuenta sus eventos, adaptadores de recursos, funcionamiento, etc. De esta manera se obtiene una descripción estándar para todos los SBB que se piensan utilizar [62].

En la siguiente tabla se propone una plantilla para descripción de SBB, explicando el contenido de cada campo.

Característica	Descripción
Nombre de SBB	El nombre del SBB
Proveedor	Quien desarrolla el SBB
Versión	La versión de implementación del SBB
Descripción	Una breve descripción del funcionamiento del SBB
Nombre clase abstracta	Nombre de la clase principal en la cual se implementa la lógica del SBB.
Nombre interfaz local	Nombre de la interfaz local en caso de que exista, en donde se publican los métodos o eventos a los que otro SBB tiene acceso. Cuales son dichos métodos.
Campos CMP (Container Managed Persistent)	Nombre y clase de los campos CMP que tiene el SBB, en caso de que existan.
Nombre especificación de perfil	Cuál es el nombre de la especificación de

	perfil que utiliza, y que datos contiene cada perfil.
SBB hijos	A cuales SBB hace referencia como SBB hijos y cuál es la prioridad de estos.
Eventos	Que eventos maneja o genera el SBB. Describiendo brevemente que hace el SBB cuando se presenta cada evento.
Métodos	En caso de que contenga métodos, se describen brevemente.
Adaptadores de recursos	Con que adaptadores de recursos tiene comunicación. Cuál es la interfaz de actividad de contexto de cada adaptador de recursos, y el enlace de la entidad del adaptador.
Alarmas	Si el SBB tiene alarmas, se describe brevemente su funcionamiento.

Tabla 2. *Especificación de SBB(Propia).*

4.2.4 Composición de SBB

Como se menciona en la sección 4.2.2, este trabajo de grado se enfoca en el desarrollo y composición de servicios JAIN SLEE, y así como la identificación y especificación de servicios se centra en la identificación y especificación de SBB, esta sección será orientada a la composición de SBB según como se encuentra definido en la especificación JAIN SLEE.

La composición de SBB se lleva a cabo a través del establecimiento de relaciones entre los SBB. Según la especificación JAIN SLEE es posible establecer relaciones entre SBB de forma síncrona y asíncrona. A continuación se describe la forma como se lleva a cabo la composición de SBB tanto síncrona como asíncronamente.

4.2.4.1 Composición Síncrona

Se le llama composición síncrona debido a que existe una relación directa entre los SBB involucrados en la composición y el proceso lógico que se lleva a cabo entre los SBB para cumplir una tarea o funcionalidad, se realiza paso a paso y con un orden lógico predeterminado. En la figura 16 se observa una composición de SBB de manera síncrona, donde el SLEE se encarga de entregar los eventos iniciales a los SBB raíces X1, X2 y X3 los cuales de forma síncrona y paso a paso invocan funciones o transfieren los eventos que acaban de recibir por parte del SLEE a los SBB con los cuales han definido relaciones.

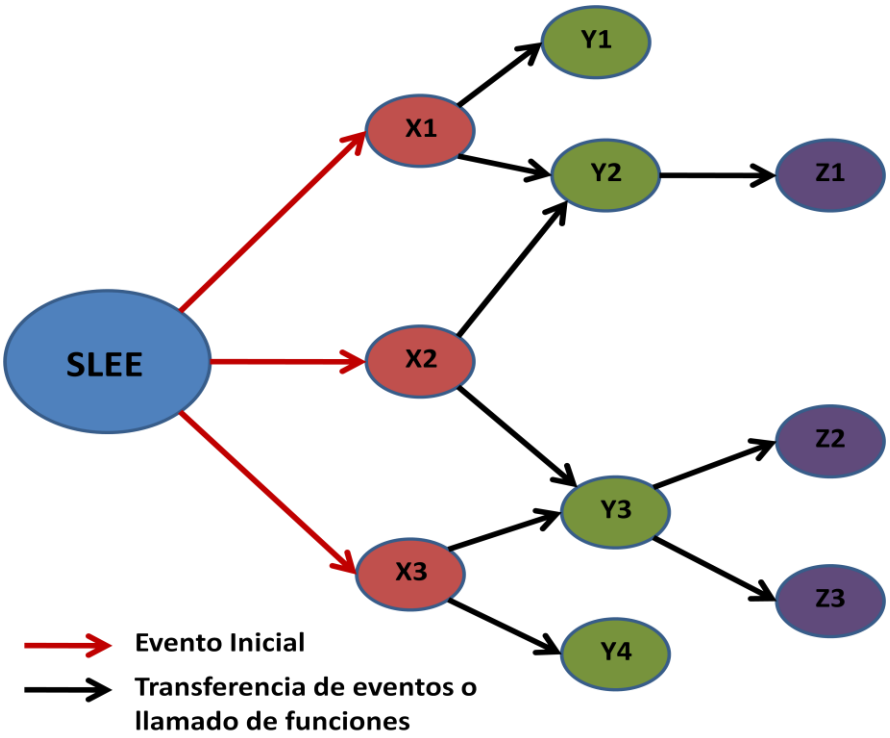

Figura 16. *Composición Síncrona de SBB (Adaptación de [3]).*

La composición síncrona de SBB se realiza a partir de la definición de una relación entre dos SBB denominada *Child Relation,* en la cual se identifica un SBB "padre" y un SBB "hijo". Se denomina "padre" al SBB principal en la relación, es decir, al SBB que utiliza o inicializa la funcionalidad de un SBB "hijo". Esta relación es siempre definida en el archivo XML que describe al SBB "padre". A continuación, en la figura 17 se presenta la sección correspondiente a la definición de la *Child Relation* del archivo XML que describe al SBB B2BUA perteneciente al servicio de prueba descrito en la sección 3.2.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE sbb-jar (View Source for full doctype...)>
- <sbb-jar>
  - <sbb>
      <description />
      <sbb-name>B2BUA SBB</sbb-name>
      <sbb-vendor>Julian Rojas</sbb-vendor>
      <sbb-version>1.0</sbb-version>
    - <sbb-ref>
        <sbb-name>Register SBB</sbb-name>
        <sbb-vendor>Julian Rojas</sbb-vendor>
        <sbb-version>1.0</sbb-version>
        <sbb-alias>RegisterSbb</sbb-alias>
      </sbb-ref>
    - <get-child-relation-method>
        <sbb-alias-ref>RegisterSbb</sbb-alias-ref>
        <get-child-relation-method-name>getRegisterSbb</get-child-relation-method-name>
        <default-priority>0</default-priority>
      </get-child-relation-method>
```

Figura 17. *Definición de una Child Relation (Propia).*

Como se observa en la figura 17, al definir una *Child Relation,* inicialmente se crea una referencia al SBB "hijo", en este caso *Register SBB* correspondiente al servicio de prueba de la sección 3.2*,* por medio de la etiqueta XML **<sbb-ref>.** Posteriormente se define el nombre del método encargado de crear la relación entre los dos SBB en tiempo de ejecución por medio de la etiqueta XML **<get-child-relation-method>**. Este método (en este caso *getRegisterSbb)* es implementado en la lógica de funcionamiento del SBB "padre" (en este caso *B2BUA SBB*). Adicionalmente es posible establecer un parámetro de prioridad de la relación lo cual es utilizado en caso de que el SBB "padre" defina más de una *Child Relation.*

En la sección 4.2.2.3 se realiza una clasificación de los SBB en compuestos y atómicos. Es posible afirmar que al definir una relación entre dos SBB, el SBB "padre" se clasifica como un SBB compuesto y el SBB "hijo" podría ser clasificado como un SBB atómico, siempre y cuando este no defina una *Child Relation* con algún otro SBB.

Una vez definida una *Child Relation* entre dos SBB, el SBB "padre" puede hacer uso de dicha relación de dos formas distintas. Una de las maneras de hacer uso de la relación entre dos SBB por parte del SBB "padre" consiste en la utilización de las capacidades y funcionalidades definidas en la lógica del SBB "hijo" a través de una interfaz denominada de forma general *[Nombre]SbbLocalObject* en la cual se referencian los métodos que serán utilizados por el SBB "padre". La otra forma de hacer uso de la relación por parte del SBB "padre", consiste en la transferencia de eventos, es decir, que cuando el SBB "padre" recibe un cierto evento, este transfiere el evento al SBB "hijo" donde el evento es procesado. Para esto es necesario que tanto el SBB "padre" como el "hijo" tengan definido el evento en sus respectivos archivos XML.

A continuación se presenta la manera de llevar a cabo, cada una de las dos formas de utilización de una *Child Relation* mencionadas anteriormente, para realizar una composición síncrona.

Interfaz SbbLocalObject:

Para la utilización de la interfaz *SbbLocalObject* por parte del SBB "padre", inicialmente esta interfaz debe ser creada y definida en el archivo XML perteneciente al SBB "hijo" como se muestra en la figura 18.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE sbb-jar (View Source for full doctype...)>
- <sbb-jar>
  - <sbb>
      <description />
      <sbb-name>Register SBB</sbb-name>
      <sbb-vendor>Julian Rojas</sbb-vendor>
      <sbb-version>1.0</sbb-version>
    - <sbb-local-interface>
        <sbb-local-interface-name>org.sip.RegisterSbbLocalObject</sbb-local-interface-name>
      </sbb-local-interface>
```

Figura 18. *Definición de Interfaz SbbLocalObject (Propia).*

En la figura 18 se observa la definición de la interfaz *SbbLocalObject* correspondiente al SBB Register que hace parte del servicio de prueba descrito en la sección 3.2. La definición de esta interfaz se realiza a través de la etiqueta XML **<sbb-local-interface-name>,** que en este caso es llamada *org.sip.RegisterSbbLocalObject* donde org.sip corresponde al nombre del paquete Java en donde se encuentra la interfaz.

Una vez definida la interfaz, es necesario incluir los métodos que serán utilizados por el SBB "padre" como se muestra en la figura 19:

```
package org.sip;

import javax.sip.header.ContactHeader;

public interface RegisterSbbLocalObject extends javax.slee.SbbLocalObject {

        public void Register(ContactHeader contactHeader);
        public String VerifyUri(ToHeader toHeader);
        public String VerifyName(ToHeader toHeader);

}
```

Figura 19. *Contenido de Interfaz SbbLocalObject (Propia).*

En la figura 19 se observa la implementación de la interfaz *SbbLocalObject* perteneciente al SBB Register del ejemplo de prueba descrito en la sección 3.2 y cuya definición se observa en la figura 18. Como se mencionó anteriormente, en esta interfaz se referencian los métodos públicos del SBB Register y a través de los cuales, el SBB B2BUA almacena y obtiene de la especificación de perfil, la información de contacto de los usuarios del servicio.

Finalmente, para utilizar dichos métodos y funcionalidades desde el SBB "padre", en este caso el SBB B2BUA es necesario crear una instancia de la interfaz a través de la cual se utilizaran los métodos allí referenciados. En la figura 20 se observa como a través del método **getRegisterSbb(),** definido en el archivo XML que describe al SBB B2BUA y que se observa en la figura 17, se crea un objeto de la clase **ChildRelation**. Posteriormente, se crea la instancia de la interfaz *SbbLocalObject* por medio del método **childRelation.create()** y es a través de esta instancia que se hace uso de los métodos definidos en la interfaz, observados en la figura 19.

```
ChildRelation childRelation = this.getRegisterSbb();
RegisterSbbLocalObject registerSbb = (RegisterSbbLocalObject) childRelation.create();
registerSbb.Register(contactHeader);
String profileUri = registerSbb .VerifyUri(toHeader);
String profileName = registerSbb.VerifyName(toHeader);
```

Figura 20. *Utilización de Interfaz SbbLocalObject (Propia).*

Transferencia de Eventos:

Como se explicó anteriormente, la trasferencia de eventos consiste en que el SBB "padre" transfiere un evento que ha recibido hacia el SBB "hijo" donde es procesado. Para esto es necesario que el evento esté definido en los archivos XML tanto del SBB "padre" como del SBB "hijo".

Para ilustrar la transferencia de eventos se hace uso de un servicio simple, el cual se compone de dos SBB llamados Child SBB y Parent SBB. Este servicio consiste en la recepción de un evento de registro SIP por parte del Parent SBB el cual transfiere el evento hacia el Child SBB haciendo uso de la transferencia de eventos. En este servicio el SBB "padre" es el Parent SBB y el SBB "hijo" es el Child SBB.

En las figuras 21 y 22 se observan ejemplos de la definición del evento tanto en el Parent SBB, como en el Child SBB.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE sbb-jar (View Source for full doctype...)>
- <sbb-jar>
  - <sbb>
      <description />
      <sbb-name>Child SBB</sbb-name>
      <sbb-vendor>Julian Rojas</sbb-vendor>
      <sbb-version>1.0</sbb-version>
    - <event event-direction="Receive" initial-event="False" mask-on-attach="False">
        <event-name>REGISTER</event-name>
      - <event-type-ref>
          <event-type-name>javax.sip.message.Request.REGISTER</event-type-name>
          <event-type-vendor>net.java.slee</event-type-vendor>
          <event-type-version>1.2</event-type-version>
        </event-type-ref>
      </event>
```

Figura 21. *Definición de Evento en SBB hijo (Propia).*

La figura 21 presenta un fragmento del archivo de descripción XML del Child SBB, en donde por medio de la etiqueta XML **<event>** se define el evento de registro SIP. La etiqueta **<event-type-ref>** es la referencia del evento de registro SIP, de la forma como se encuentra definido en el adaptador de recursos SIP. Vale la pena resaltar que el parámetro **initial-event** perteneciente a la etiqueta XML **<event>,** se encuentra definido como **"False",** debido a que el evento de registro SIP no es un evento de tipo inicial para el Child SBB dentro de la lógica del servicio. A diferencia del método de composición síncrona por medio de la interfaz *SbbLocalObject,* en la transferencia de eventos no es necesario definir esta interfaz para los SBB "hijos", debido a que los SBB padres no hacen uso de los métodos definidos por los SBB "hijos".

De la misma forma en cómo se encuentra definido el evento de registro SIP en el Child SBB en la figura 21, se define este evento en el Parent SBB exceptuando al parámetro **initial-event** el cual se encuentra definido como **"True"** ya que este evento si representa un evento inicial para la lógica del servicio. Esta definición se observa en la figura 22.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE sbb-jar (View Source for full doctype...)>
- <sbb-jar>
  - <sbb>
      <description />
      <sbb-name>Parent SBB</sbb-name>
      <sbb-vendor>Julian Rojas</sbb-vendor>
      <sbb-version>1.0</sbb-version>
    - <sbb-ref>
        <sbb-name>Child SBB</sbb-name>
        <sbb-vendor>Julian Rojas</sbb-vendor>
        <sbb-version>1.0</sbb-version>
        <sbb-alias>ChildSbb</sbb-alias>
      </sbb-ref>
    - <get-child-relation-method>
        <sbb-alias-ref>ChildSbb</sbb-alias-ref>
        <get-child-relation-method-name>getChildSbb</get-child-relation-method-name>
        <default-priority>0</default-priority>
      </get-child-relation-method>
    - <event event-direction="Receive" initial-event="True" mask-on-attach="False">
        <event-name>REGISTER</event-name>
      - <event-type-ref>
          <event-type-name>javax.sip.message.Request.REGISTER</event-type-name>
          <event-type-vendor>net.java.slee</event-type-vendor>
          <event-type-version>1.2</event-type-version>
        </event-type-ref>
        <initial-event-select variable="ActivityContext" />
      </event>
```

Figura 22. *Definición de Evento en SBB padre (Propia).*

En la figura 22, aparte de la definición del evento de registro SIP, también se observa la referencia hacia el ChildSBB en la etiqueta XML **<sbb-ref>,** y la definición del nombre del método encargado de crear la relación entre los dos SBB en tiempo de ejecución por medio de la etiqueta XML **<get-child-relation-method>.**

Una vez que se ha definido el evento tanto para el Parent SBB (SBB “padre”) como para el Child SBB (SBB “hijo”), es necesario relacionar al Child SBB con la actividad de contexto del evento que se quiere transferir, en este caso el evento de registro SIP. Para esto, inicialmente se crea un objeto de la clase **ChildRelation** por medio del método **this.getChildSBB()** definido en el archivo de descripción XML de la figura 22. Posteriormente se crea un objeto que representa una instancia del Child SBB con el método **childRelation.create().** Finalmente se crea la relación entre la actividad de contexto del evento con la instancia del Child SBB por medio del método **aci.attach(child).** Este proceso se realiza dentro del método que maneja el evento de registro SIP en el Parent SBB. En la figura 23 se observa este procedimiento.

```
public void onREGISTER(RequestEvent event, ActivityContextInterface aci) {
        System.out.println("Entra al metodo onREGISTER del ParentSBB");
        ChildRelation childRelation = this.getChildSbb();
        try {
                SbbLocalObject child = childRelation.create();
                //Metodo necesario para redirigir el evento hacia un child SBB
                aci.attach(child);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

Figura 23. *Transferencia de Evento (Propia).*

Es mediante los dos métodos anteriormente descritos que es posible realizar composición de SBB de forma síncrona.

4.2.4.2 Composición Asíncrona

Es llamada composición asíncrona debido a que a diferencia de la composición síncrona, no existe una relación directa entre los SBB involucrados en la composición y no se define una conexión a nivel de código ni en los archivos descriptores XML de los SBB. Otra característica de la composición asíncrona es que no se hace una distinción entre los SBB como en la composición síncrona, donde en la definición de relaciones, se distingue entre SBB “padres” y SBB “hijos”. Cabe aclarar que sin importar el tipo de composición que se utilice, la especificación JAIN SLEE define un SBB “raíz” para cada servicio que se desarrolle bajo la especificación, a través del cual se inicializa el servicio. Por lo tanto en la composición asíncrona también existe un SBB “raíz” para cada servicio pero en la relación de la lógica de funcionamiento que existe entre los SBB que participan en el servicio no se realiza ninguna clasificación.

La composición asíncrona de SBB consiste en la recepción y disparo de eventos a través del SLEE, es decir, que las relaciones a nivel de lógica de funcionamiento entre SBB se realizan a partir de la recepción de un evento en un SBB, el cual procesa dicho evento y puede proceder a disparar otro evento que es recibido por otro SBB. Este proceso puede ser realizado cuantas veces sea necesario dentro de un servicio para cumplir una tarea o funcionalidad. Cabe resaltar que este proceso supone un nivel más alto de procesamiento en comparación con la composición síncrona, debido a las tareas de recepción, enrutamiento y disparo de eventos que se realizan dentro del SLEE.

En la figura 24 se presenta un ejemplo de composición asíncrona de SBB, donde el SBB X es el SBB “raíz” del servicio y por lo tanto cuando recibe un evento inicial, se da inicio a la lógica del servicio. A partir de la recepción del evento inicial por parte del SBB X, este procede a disparar otros eventos al SLEE que son recibidos por los otros SBB pertenecientes al servicio, y que a su vez tienen la capacidad seguir disparando eventos en el SLEE. Como se mencionó anteriormente, este proceso puede repetirse indefinidamente hasta lograr el objetivo del servicio.

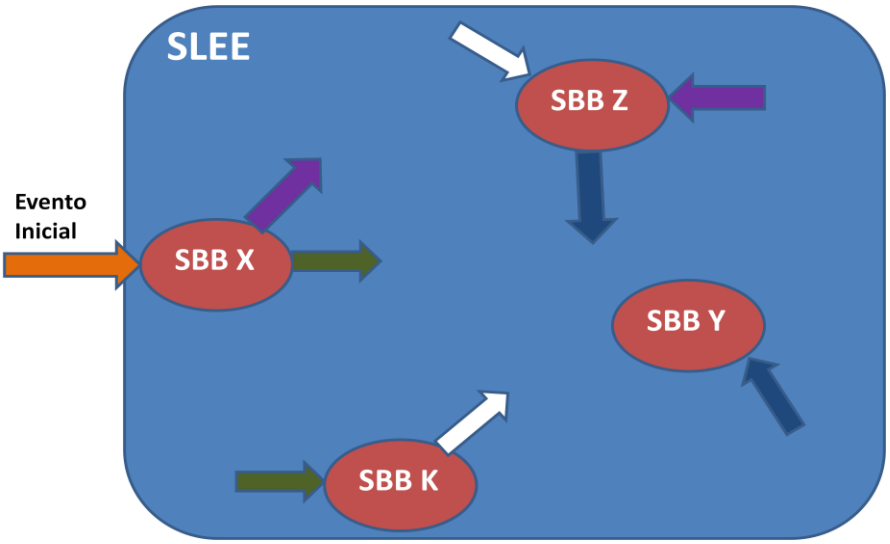

Figura 24. *Composición Asíncrona de SBB (Propia).*

Debido a que en la composición asíncrona de SBB no se define una relación a nivel de código entre SBB, para llevarla a cabo simplemente basta con definir los eventos que se necesita que sean recibidos y disparados en cada uno de los SBB involucrados en la composición. En la figura 25 se observa la configuración para recibir y disparar eventos en un archivo descriptor XML de un SBB llamado Parent SBB.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE sbb-jar (View Source for full doctype...)>
- <sbb-jar>
  - <sbb>
      <description />
      <sbb-name>Parent SBB</sbb-name>
      <sbb-vendor>Julian Rojas</sbb-vendor>
      <sbb-version>1.0</sbb-version>
    - <event event-direction="Receive" initial-event="True" mask-on-attach="False">
        <event-name>REGISTER</event-name>
      - <event-type-ref>
          <event-type-name>javax.sip.message.Request.REGISTER</event-type-name>
          <event-type-vendor>net.java.slee</event-type-vendor>
          <event-type-version>1.2</event-type-version>
        </event-type-ref>
        <initial-event-select variable="ActivityContext" />
      </event>
    - <event event-direction="Fire" initial-event="True" mask-on-attach="False">
        <event-name>InviteEvent</event-name>
      - <event-type-ref>
          <event-type-name>javax.sip.message.Request.INVITE</event-type-name>
          <event-type-vendor>net.java.slee</event-type-vendor>
          <event-type-version>1.2</event-type-version>
        </event-type-ref>
```

Figura 25. *Definición de Eventos (Propia).*

En la figura anterior se presenta el archivo descriptor XML de un SBB llamado Parent SBB, para el cual se ha definido que puede recibir eventos de registro SIP y puede disparar eventos SIP del tipo Invite. La diferencia en la definición de los eventos que hacen referencia a sin son recibidos o disparados por el SBB se da en la etiqueta XML **<event>,** en la cual se define el parámetro **event-direction** como **"Fire"** para disparar eventos, o como **"Receive"** para recibirlos.

Es de esta forma que se puede llevar a cabo la composición asíncrona de servicios JAIN SLEE.

4.2.5 Implementación del Servicio

El último paso que se plantea dentro de estos lineamientos para composición de servicios JAIN SLEE consiste en la implementación del servicio, el cual es el producto final obtenido a partir de la realización de todos los pasos propuestos en los lineamientos. En esta sección se presenta el proceso general necesario para la implementación de un servicio JAIN SLEE, a través de las herramientas EclipSLEE y Alcatel-Lucent (SCE-SE), propuestas como la mejor alternativa en el capítulo 3.

A partir de la utilización de estas herramientas, los pasos a seguir para la implementación de un servicio JAIN SLEE en general, son los siguientes:

Creación del Proyecto del Servicio:

Este es el primer paso en la implementación del servicio y consiste en crear un proyecto JAIN SLEE el cual es una capacidad que provee EclipSLEE. Aquí se le da un nombre al proyecto y se especifica el lugar donde será creado. En la figura 26 se observa la opción que proporciona EclipSLEE para crear un proyecto JAIN SLEE.

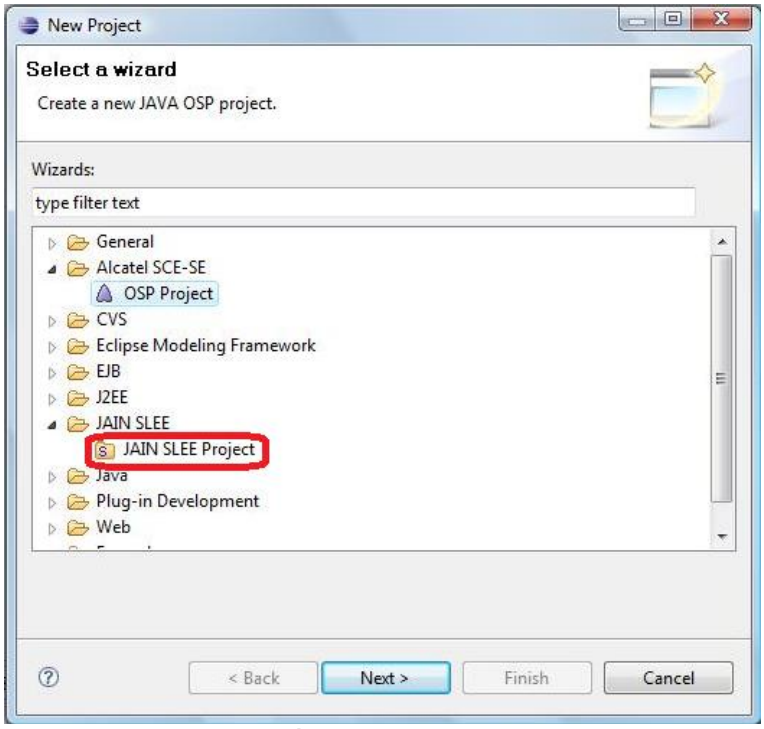

Figura 26. *Creación de Proyecto JAIN SLEE* [46].

Creación de Eventos y Especificaciones de Perfil:

Este es un paso opcional ya que no todos los servicios que se desarrollan hacen uso de eventos personalizados o creados por el desarrollador, ni de especificaciones de perfil. Sin embargo, en caso de que el servicio a implementar requiera de estos componentes JAIN SLEE, es necesario que sean implementados antes de que cualquier SBB que los utilice. Por otra parte, debido a que tanto EclipSLEE como Alcatel-Lucent poseen la capacidad de crear eventos personalizados y especificaciones de perfil, es indiferente cual de las dos herramientas sea utilizada en estos procesos. En la figura 27 se observa el menú de Alcatel-Lucent para la creación de estos componentes.

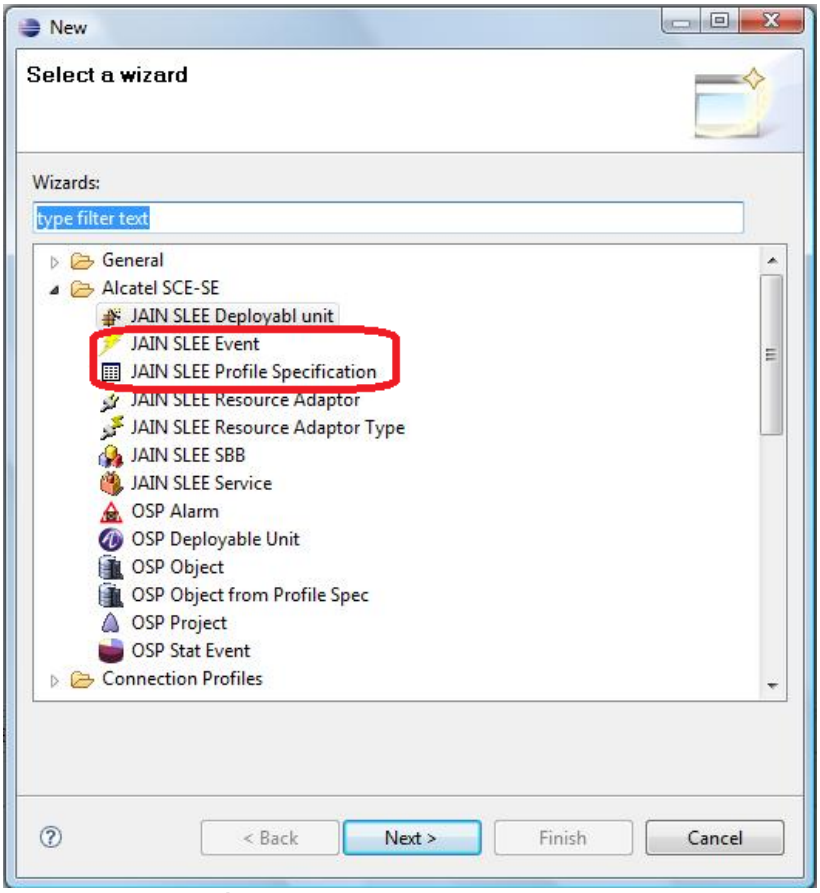

Figura 27. *Creación de Eventos y Especificaciones de Perfil* [47].

Tanto para la creación de eventos como de especificaciones de perfil, es necesario darles un nombre y especificar el paquete Java donde serán almacenados, pero adicionalmente, para las especificaciones de perfil se deben crear los campos CMP (Container Managed Field) que sean necesarios. Estos campos se encargan de almacenar la información de los diferentes perfiles. En la figura 28 se observa la interfaz que permite crear los campos CMP para las especificaciones de perfil.

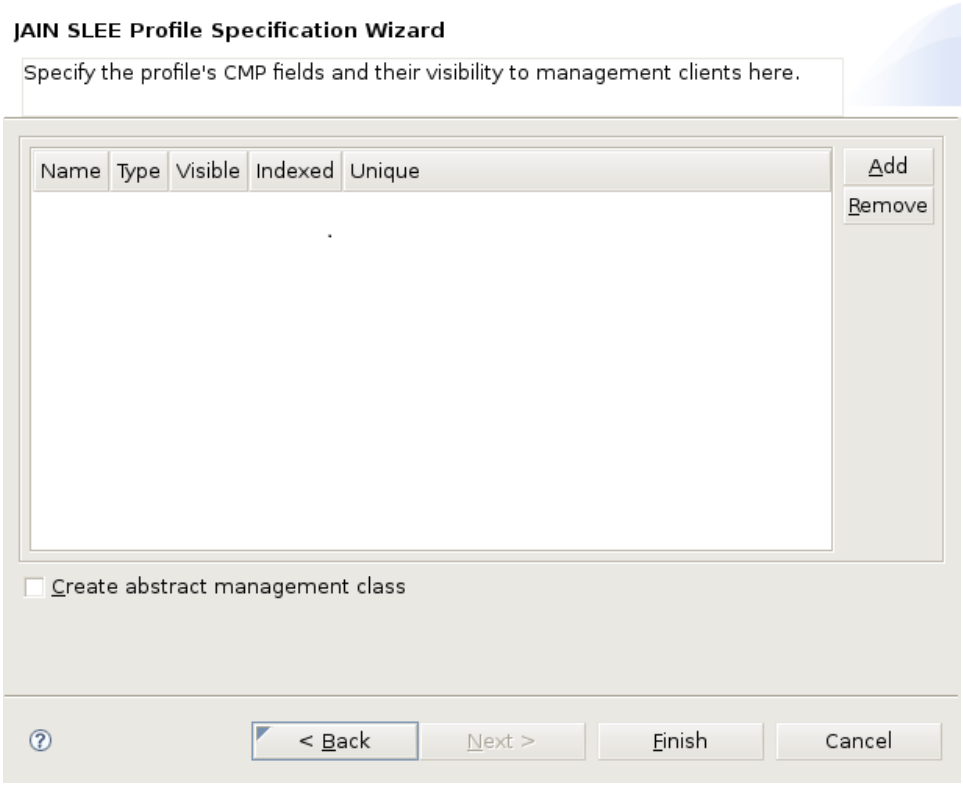

Figura 28. *Creación de campos CMP para Especificaciones de Perfil* [46].

Creación de SBB:

Se puede afirmar que este es el paso principal dentro del proceso de implementación de un servicio JAIN SLEE ya que son los SBB quienes contienen la lógica de funcionamiento del servicio. Para la creación de los SBB, es necesario tener presente el tipo de composición que se va a utilizar ya que en caso de ser una composición síncrona es indispensable el orden en el cual los SBB son creados, siendo los SBB atómicos los primeros en ser creados y por último los SBB compuestos (Sección 4.2.2.3). En caso de utilizarse una composición asíncrona es indiferente el orden en el cual sean creados los SBB.

Los SBB deben ser creados utilizando las capacidades de la herramienta Alcatel-Lucent debido a que es esta herramienta la que provee la funcionalidad de la máquina de estados con interfaz gráfica para los SBB. Se debe tener en cuenta también, que si es necesaria la utilización de la interfaz *SbbLocalObject* de algún SBB, esta debe ser habilitada por el desarrollador en las opciones de creación del SBB.

Durante el proceso de creación de un SBB, inicialmente se debe especificar su nombre y el paquete Java donde será almacenado. Después se determina si debe ser creada la interfaz *SbbLocalObject* y se da la opción de asignarle un nombre (figura 29). Posteriormente, se crean los campos CMP del SBB en caso de ser necesarios (figura 30). Seguido a esto se definen los diferentes eventos que el SBB tendrá la capacidad de procesar, ya sea recibiéndolos o disparándolos (figura 31). El siguiente paso consiste en referenciar los SBB hijos con los cuales se relaciona el SBB en caso de tener alguno (figura 32). Finalmente se escoge el tipo de arquitectura que se quiere que tenga el SBB

ya sea la tradicional o la de máquina de estados que ofrece la herramienta Alcatel-Lucent (figura 33).

A continuación se presentan las diferentes interfaces que permiten la creación de un SBB con la herramienta Alcatel-Lucent.

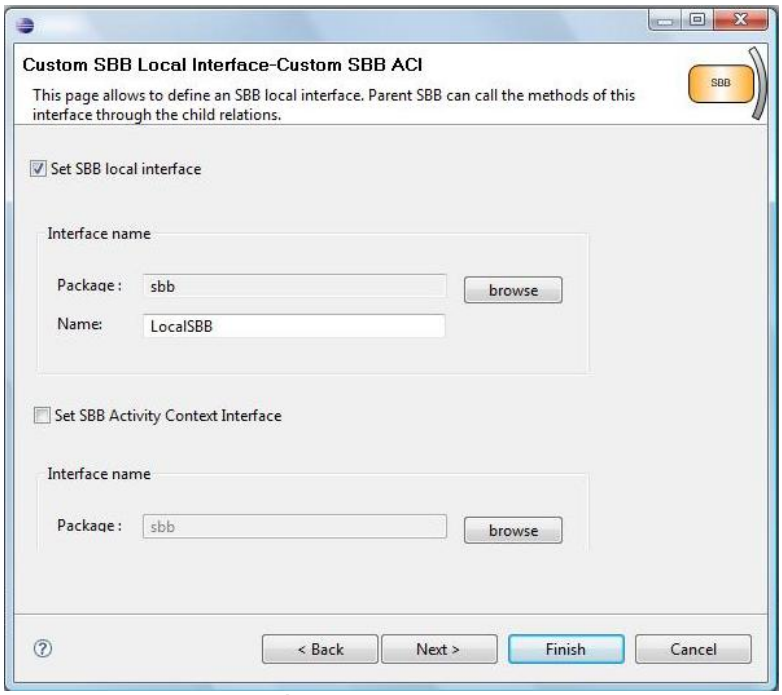

Figura 29. *Creación de Interfaz SbbLocalObject* [47].

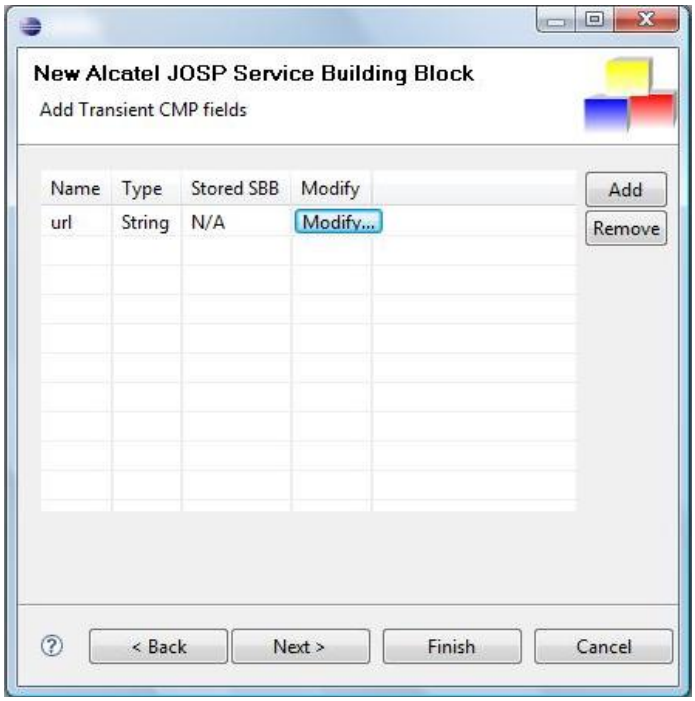

Figura 30. *Interfaz para Creación de campos CMP de un SBB* [47].

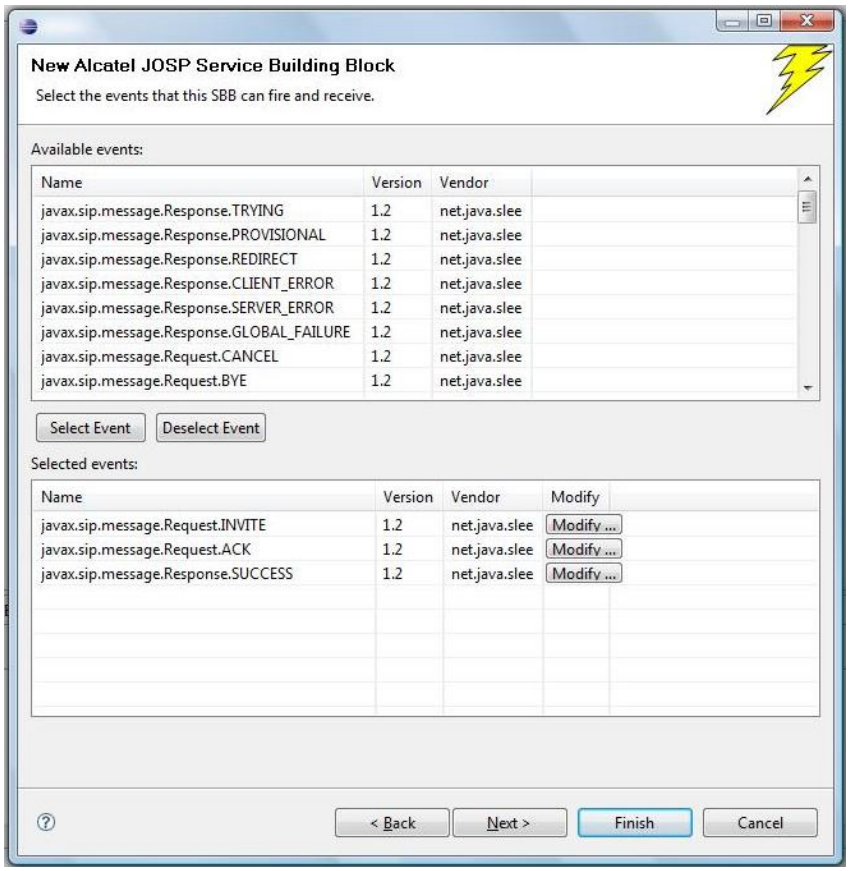

Figura 31. *Interfaz para Asignación de Eventos de un SBB* [47].

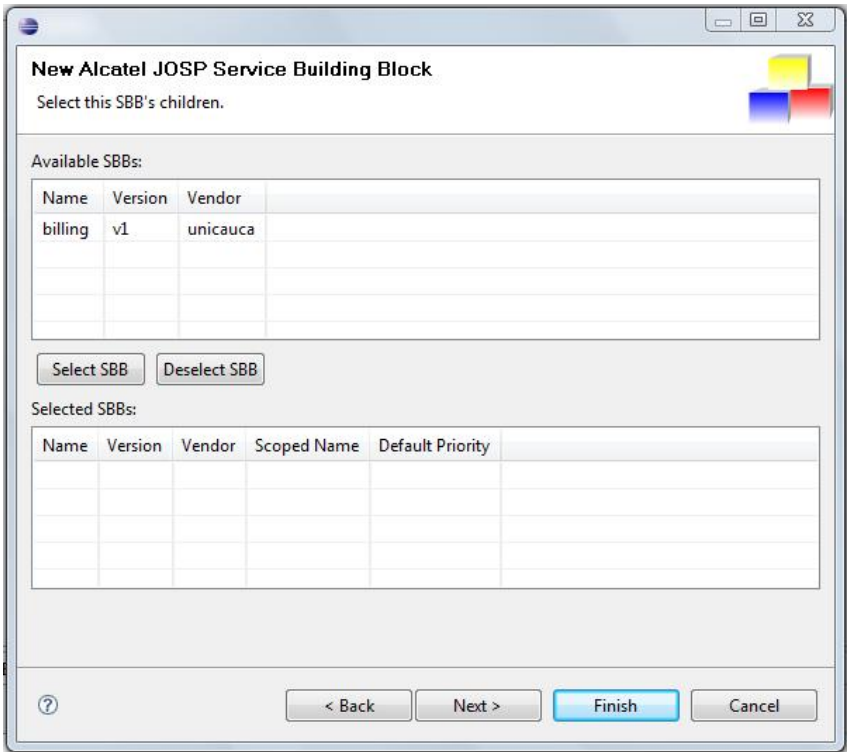

Figura 32. *Interfaz para Asignación de SBB hijos* [47].

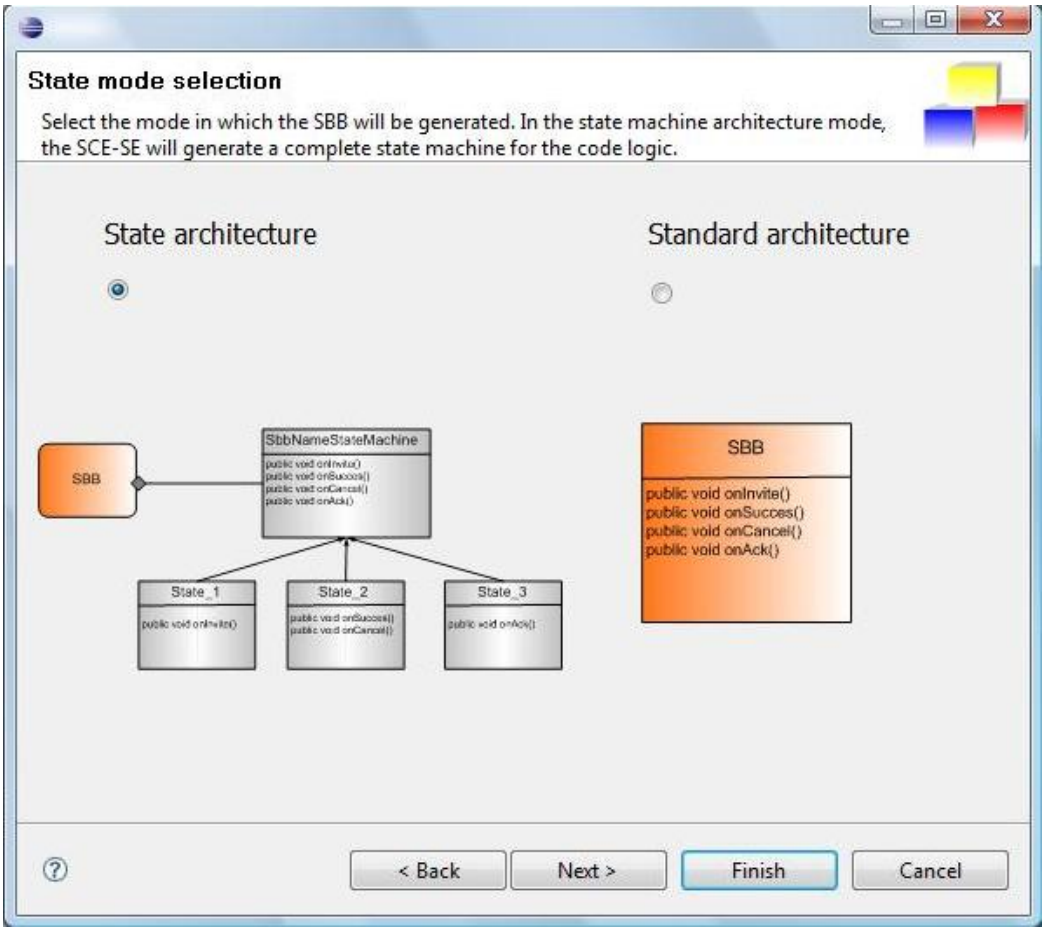

Figura 33. *Interfaz de tipo de Arquitectura de un SBB* [47].

Creación de Descriptor de Servicio y Unidad Desplegable:

El descriptor de servicio es un archivo XML en el cual se hace referencia al SBB "raíz" del servicio, al nombre del servicio y a la prioridad del servicio dentro del SLEE. La unidad desplegable, es el archivo JAR en el cual se encuentran empaquetados todos los componentes JAIN SLEE que conforman el servicio. Para la creación de estos componentes, se utiliza la herramienta EclipSLEE, ya que Alcatel-Lucent incluye algunas características propietarias que hacen referencia a su Plataforma Abierta de Servicios (OSP) y que no pertenecen a la especificación JAIN SLEE. Por último, es necesario crear primero el descriptor de servicio antes que la unidad desplegable, ya que en la unidad desplegable se hace referencia al descriptor de servicio.

Despliegue del Servicio:

Finalmente, una vez creados todos los componentes necesarios en el servicio mediante los pasos anteriormente descritos y obtenida la unidad desplegable del servicio, el último paso restante es el despliegue del servicio en el SLEE. Esto es posible gracias a la funcionalidad que provee EclipSLEE de despliegue directo del servicio en un SLEE local. Sin embargo, los SLEE como Mobicents poseen interfaces de gestión que permiten realizar el despliegue de los servicio. A continuación se observa la capacidad de despliegue de servicios que provee EclipSLEE:

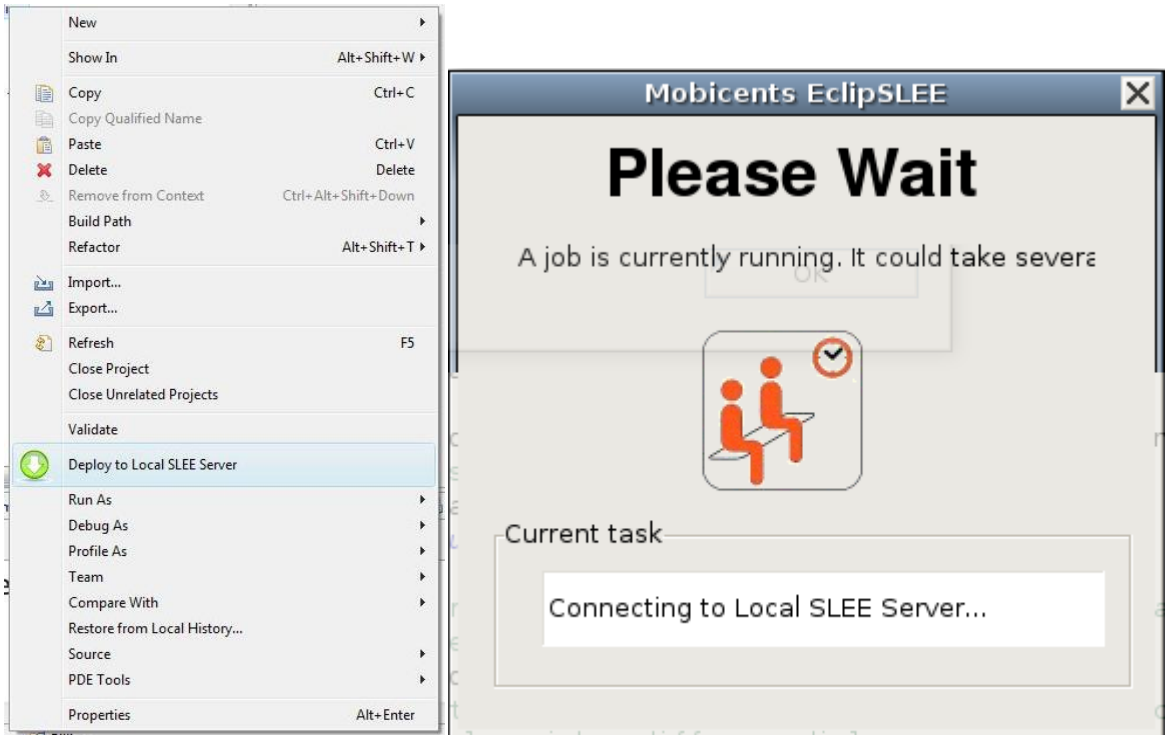

Figura 34. *Capacidad de despliegue de Servicios de EclipSLEE* [46].

RESUMEN

En este capítulo se presentaron los lineamientos para composición de servicios de telecomunicaciones JAIN SLEE propuestos, en donde se llevan a cabo una serie de pasos que parten desde el modelamiento de la idea de negocio basado en el modelo para la creación de servicios de SOA, en donde se realiza un análisis de la idea con el fin de establecer unos objetivos y requisitos necesarios para la construcción de una solución que satisfaga dichos objetivos. Posteriormente se realiza un proceso de identificación de servicios en donde se parte de los objetivos y requisitos previamente establecidos para plantear procesos y servicios que actuando de forma compuesta puedan cumplir con los objetivos propuestos. Dado que este trabajo trata la composición de servicios JAIN SLEE, la identificación de servicios hace referencia a los Bloques Constructores de Servicios (SBB) definidos por la especificación JAIN SLEE. Como paso siguiente se realiza una especificación de servicios, en donde se describe de forma detallada cada uno de los SBB's identificados en el paso anterior. Posteriormente se presentan una serie de alternativas para realizar composición de servicios basadas en la especificación JAIN SLEE, que tienen diferentes características y que se ajustan a diferentes escenarios. Finalmente se presentan una serie de pasos que permiten implementar el servicio que ha sido desarrollado por medio de los lineamientos, utilizando las herramientas especificadas en el capítulo 3.

Capítulo V

IMPLEMENTACIÓN DEL PROTOTIPO

5.1 Introducción

En este capítulo será presentada la implementación del prototipo a través del cual se validaran los lineamientos para composición de servicios de telecomunicaciones en entornos JAIN SLEE, propuestos en el capitulo anterior. Dicha implementación del prototipo consiste en el desarrollo de dos servicios de telecomunicaciones, siguiendo paso a paso los lineamientos y utilizando el entorno de desarrollo definido en el Capítulo III.

El objetivo principal de este capítulo es demostrar que al seguir paso a paso los lineamientos, es posible obtener robustos servicios convergentes en tiempos muy cortos, debido a que los lineamientos proponen un método de desarrollo basado en la composición y reutilización de componentes JAIN SLEE.

5.2 Implementación del Prototipo

La implementación de los servicios se llevara a cabo siguiendo la estructura que plantea los lineamientos, por lo tanto inicialmente será realizado el modelado del negocio, luego la identificación y especificación de los servicios, y finalmente la implementación de estos.

5.2.1 Modelado del Negocio

5.2.1.1 Visión del Negocio

Actualmente los servicios de información tales como e-mail, redes sociales y aplicaciones web, tienen un gran auge en nuestra sociedad, al igual que los servicios tradicionales de telefonía fija y móvil. Por lo tanto los proveedores de telecomunicaciones buscan integrar estos dos tipos de servicios, y de esta manera ofrecer servicios convergentes y ampliar su portafolio de servicios.

Teniendo en cuenta lo anterior, se define como objetivo principal del negocio desarrollar dos servicios convergentes, los cuales serán ofrecidos por parte de algún proveedor de telecomunicaciones que cuente con redes de nueva generación. Estos servicios estarán enfocados en mejorar la eficiencia y productividad del área comercial de las empresas, proporcionando la capacidad de realizar llamadas a los clientes de forma automática y dotando de ubicuidad a su personal. Además la empresa tendrá la posibilidad de gestionar los servicios por medio de interfaces Web.

El primer servicio llamado *Numero Portable*, consiste en proporcionar a los usuarios un número telefónico único a través del cual puedan ser ubicados sin importar cual sea el número real en el cual se encuentren disponibles. Por ejemplo, un usuario tiene como numero portable el numero “100” y actualmente se encuentra disponible en el numero “101”, por lo tanto a través del servicio de *Numero Portable*, cada vez que se realice una llamada al número “100” esta será redirigida hacia el numero “101”, permitiéndole al

usuario ser ubicado por medio de su número portable sin importar en cual numero se encuentre disponible.

El segundo servicio conocido como *Llamada Recordatoria,* consiste en proporcionar la capacidad de definir a través de una interfaz Web una lista de números telefónicos a los cuales se requiere llamar para entregarles un determinado mensaje, lo cual es realizado de forma automática por el servicio. Por ejemplo, una empresa que maneja un departamento de cartera tiene una lista de clientes morosos y requiere llamarlos para recordarles sus respectivos pagos. Por medio del servicio *Llamada Recordatoria* es posible realizar las llamadas a todos los usuarios pertenecientes a la lista y recordarles sus respectivos pagos con un mensaje predeterminado de forma automática.

5.2.1.2 Casos de Uso del Negocio

Basándose en la visión del negocio y en el tipo de servicios que se requieren, a continuación son definidos los casos de uso del negocio. Para representar dichos casos de uso es utilizada la notación de UML, por lo tanto en esta sección se mostrará el diagrama de casos de uso del negocio, y posteriormente su respectiva explicación.

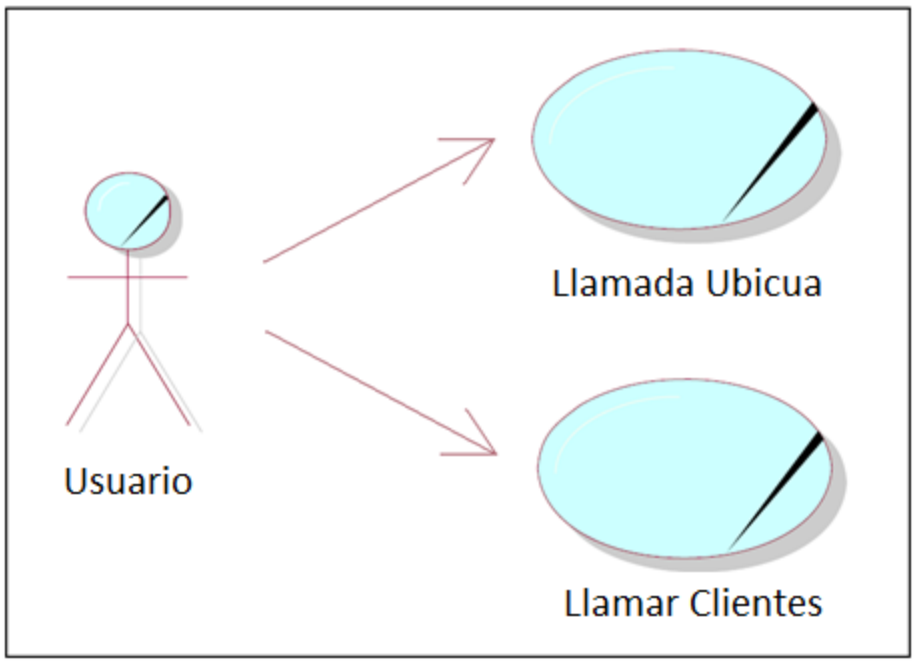

Figura 35. *Diagrama de casos de uso del Negocio (propia).*

En la figura 35, se pueden observar los dos casos de uso del negocio que representan a los dos servicios donde el caso de uso Llamada Ubicua representa al servicio *Número Portable* y el caso de uso Llamar Clientes representa al servicio *Llamada Recordatoria.* En las tablas 3 y 4 se presenta su respectiva descripción

.

Nombre caso de uso	Llamada Ubicua
Iniciador	Usuario
Propósito	Permitir al usuario realizar una llamada a otro usuario del sistema, ingresando un único número, independientemente del número en el cual el otro usuario se encuentre disponible.
Descripción	Antes de iniciar este caso de uso el usuario debe registrar un número principal y otros números alternos de diferentes ubicaciones. Inicialmente un usuario “A” ingresa al sistema desde cualquier ubicación y el sistema obtiene y almacena el número de dicha ubicación y cuando un usuario “B” intenta realizar una llamada al usuario “A” digitando su número principal, el sistema verifica en que numero alterno se encuentra disponible “A” y redirige la llamada a ese número, el cual corresponde a la ubicación desde donde se registró el usuario “A”. Cada llamada que se realice mediante el sistema deberá quedar registrada con su respectiva fecha y duración.

Tabla 3. *Descripción de caso de uso del negocio Llamada Ubicua (propia).*

Nombre caso de uso	Llamar Clientes
Iniciador	Usuario
Propósito	Permitir al usuario crear la lista de llamadas que el sistema deba realizar.
Descripción	El usuario primero se identifica ante el sistema (con su login y password) por medio de una interfaz Web, luego ingresa una lista de contactos (digitando su número telefónico), una vez envía la lista al sistema, este la almacena; posteriormente el sistema toma cada contacto de la lista e intenta establecer una llamada; en el caso de que la llamada sea contestada el sistema reproduce una grabación con un mensaje previamente entregado por parte del usuario. Al finalizar este caso de uso el usuario podrá verificar mediante la interfaz web, un registro de información del las llamadas, especificando si la llamada fue contestada o no se pudo realizar.

Tabla 4. *Descripción de caso de uso del negocio Llamar Clientes (propia).*

5.2.1.3 Arquitectura del Negocio

Para definir la arquitectura del negocio se tiene en cuenta la descripción de los casos de uso del negocio, de esta descripción se debe identificar los componentes del sistema con los que sea posible implementar la funcionalidad requerida.

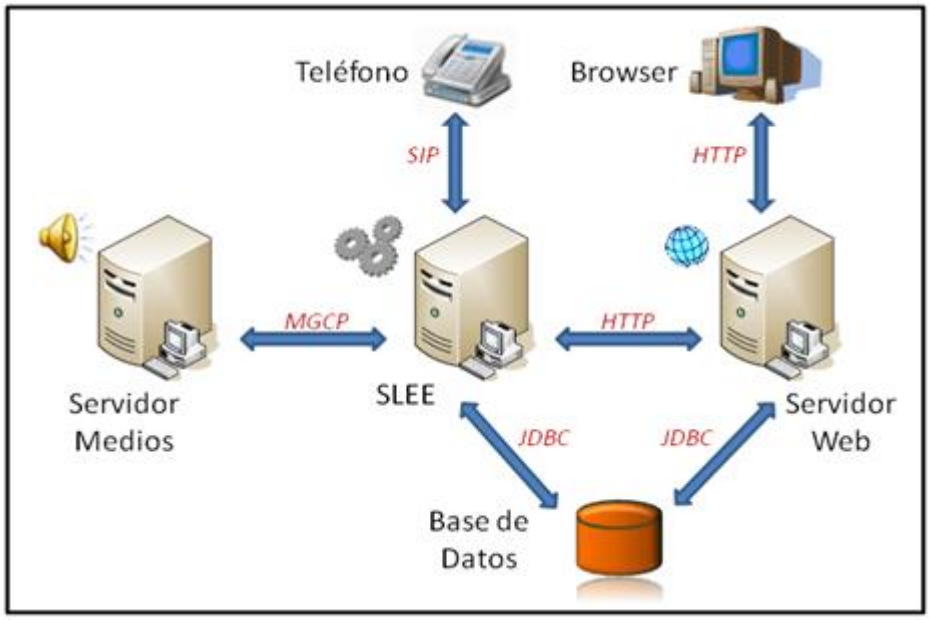

Figura 36. *Arquitectura del Negocio (propia).*

En la figura 36 se observan los componentes del sistema, dentro de los cuales existen 3 servidores, un servidor Web, un servidor de medios y un SLEE, además existe el teléfono SIP y el Browser como clientes del sistema. El componente principal de la arquitectura es el SLEE, puesto que este gestiona el acceso y ciclo de vida de los servicios, y además controla su lógica. El SLEE seleccionado para esta arquitectura es Mobicents, un SLEE robusto de libre distribución el cual fue presentado en la sección 2.1.3.3.

La selección de los demás componentes del sistema se definió de la siguiente manera: como servidor Web se definió a Glassfish v3 [63], un servidor de aplicaciones Web de libre distribución; como servidor de medios a Mobicents Media Server versión Standalone, presentado en la sección 2.1.3.3; como teléfono SIP a X-lite v3 [64]; para persistencia se utiliza una base de datos Postgres v8.3 [65]. Finalmente vale la pena resaltar que todos los componentes del sistema son ejecutados sobre el sistema operativo Ubuntu 10.04 el cual es una distribución libre de Linux.

5.2.1.4 Identificación de Objetivos y Requisitos del Negocio

Basándose en la visión, los casos de uso y la arquitectura del negocio se definieron los siguientes objetivos y requerimientos:

Objetivos:

- Desarrollar los servicios *Número Portable* y *Llamada Recordatoria* para mejorar la eficiencia y productividad del área comercial de una empresa.
- Automatizar el proceso de llamado de los clientes, que realiza la empresa con fines informativos.
- Ofrecer al personal de la empresa una mayor ubicuidad, transfiriendo sus llamadas a la ubicación que desee sin cambiar de número.

Requerimientos:

- Interfaz Web que permita obtener una lista de contactos a los cuales el sistema llamará.
- Interfaz Web que despliegue la información de las llamadas requeridas en la lista, especificando si la llamada fue exitosa o no.
- Redireccionar a los números alternos, la llamada que se le realice a un usuario.
- Almacenar un registro de cada llamada que se realice, consignando el origen, destino, fecha y duración.
- Instalar la plataforma Mobicents, la cual incluye el servidor de aplicaciones JBOSS v4.2.3 y el SLEE v1.2.7.
- Adaptadores de recursos que implementen los protocolos SIP, HTTP y MGCP.

5.2.2 Identificación de Bloques Constructores del Servicio

Como se definió en los lineamientos del Capítulo IV, la identificación de los bloques constructores del servicio pretende definir cuáles serán los SBB necesarios que implementen los requerimientos previamente definidos.

5.2.2.1 Análisis de Recursos Existentes

Al realizar el análisis de recursos existentes se encontraron algunos componentes JAIN SLEE, que dada su relación con los procesos y lógica requerida, constituyen un punto de partida que hace posible disminuir el tiempo de desarrollo mediante la reutilización de dichos componentes. Los componentes JAIN SLEE son los siguientes:

- Adaptador de recursos SIP: sip-ra-DU-1.2.7.GA
- Adaptador de recursos HTTP: http-ra-DU-1.2.7.GA
- Adaptador de recursos de Media: mgcp-ra-DU-1.0.3.GA
- SBB: SIPB2B (conecta una llamada entre 2 clientes)
- SBB: Http_servletra (recibe peticiones http e imprime el método de la solicitud)
- SBB: MGCP_callSBB (establece una sesión entre el servidor de medios y un cliente).

Estos componentes fueron tomados del grupo dirigido por la comunidad de Mobicents.

5.2.2.2 Definición de Sub-Procesos

Para la definición de subprocesos se toma la descripción de los dos casos de uso del negocio, definidos en la sección 5.2.1.2, y cada uno de ellos se descompone en sub-procesos de la siguiente manera:

Caso de Uso: Llamada Ubicua

- El usuario “A” se registra en el sistema desde cualquier ubicación.
- El sistema obtiene y almacena el número de la ubicación del usuario “A”.
- El usuario “B” intenta realizar una llamada al usuario “A”, digitando su número principal.

- El sistema verifica en que numero alterno se encuentra disponible “A” y redirige a ese numero la llamada.
- El sistema registra el origen, destino, fecha y duración de cada llamada.

Caso de Uso: Llamar Clientes

- El usuario ingresa el login y password en una interfaz Web.
- El usuario ingresa una lista de contactos en la interfaz Web.
- El sistema almacena la lista de contactos.
- El sistema intenta realizar cada llamada de la lista.
- Si el destino contesta la llamada el sistema reproduce un archivo de audio.
- El sistema guarda un registro de la realización de la llamada.
- La aplicación Web muestra el estado de las llamadas mediante una interfaz.

5.2.2.3 Definición y Clasificación de los SBB

Siguiendo las recomendaciones y teniendo en cuenta la clasificación planteada en la sección 4.2.2.3, se toman los subprocesos definidos en el literal anterior, y a partir de estos son identificados los siguientes SBB para cada de uso del negocio:

Caso de Uso: Llamada Ubicua

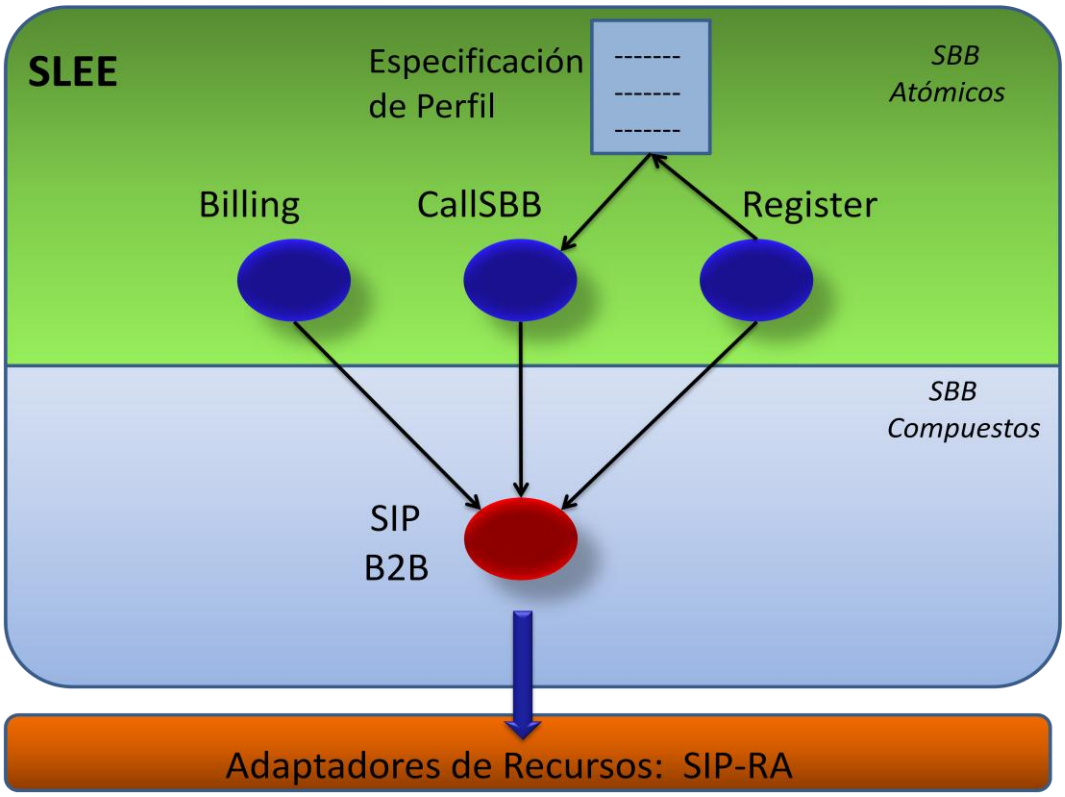

Figura 37. *Clasificación de SBB del caso de uso: Llamada Ubicua (propia).*

En la figura 37 son expuestos los SBB identificados con base en los sub-procesos definidos anteriormente. Los tres SBB atómicos definidos se encargan de funciones especificas de la siguiente manera: el SBB *Register* se encarga de obtener y almacenar en la tabla de una especificación de perfil JAIN SLEE, la URI SIP, el numero principal y el numero alterno, además guarda un indicador que determina en que numero se encuentra disponible. El *CallSBB* se encarga de extraer de la tabla la información necesaria que le permita saber el número en el cual el usuario está disponible. El SBB *Billing* es el encargado crear un registro en una base de datos con la información de la llamada. El SBB *SIP B2B* es un SBB compuesto que integra los 3 SBB atómicos e implementa todas las relaciones de composición necesarias entre estos, de tal forma que sea posible conectar una llamada entre los dos usuarios.

Caso de Uso: Llamar Clientes

Analizando los sub-procesos y requerimientos de este caso de uso, se encuentra que es necesaria una funcionalidad similar a la que implementa el caso de uso Llamada Ubicua, ya que en este caso de uso también es necesario conectar llamadas entre dos usuarios, además sería más conveniente que cuando el sistema realice las llamadas pueda encontrar a un usuario independientemente de su ubicación. Por lo tanto para definir los SBB de este caso de uso, se reutilizan todos los SBB del caso de uso Llamar de la siguiente manera:

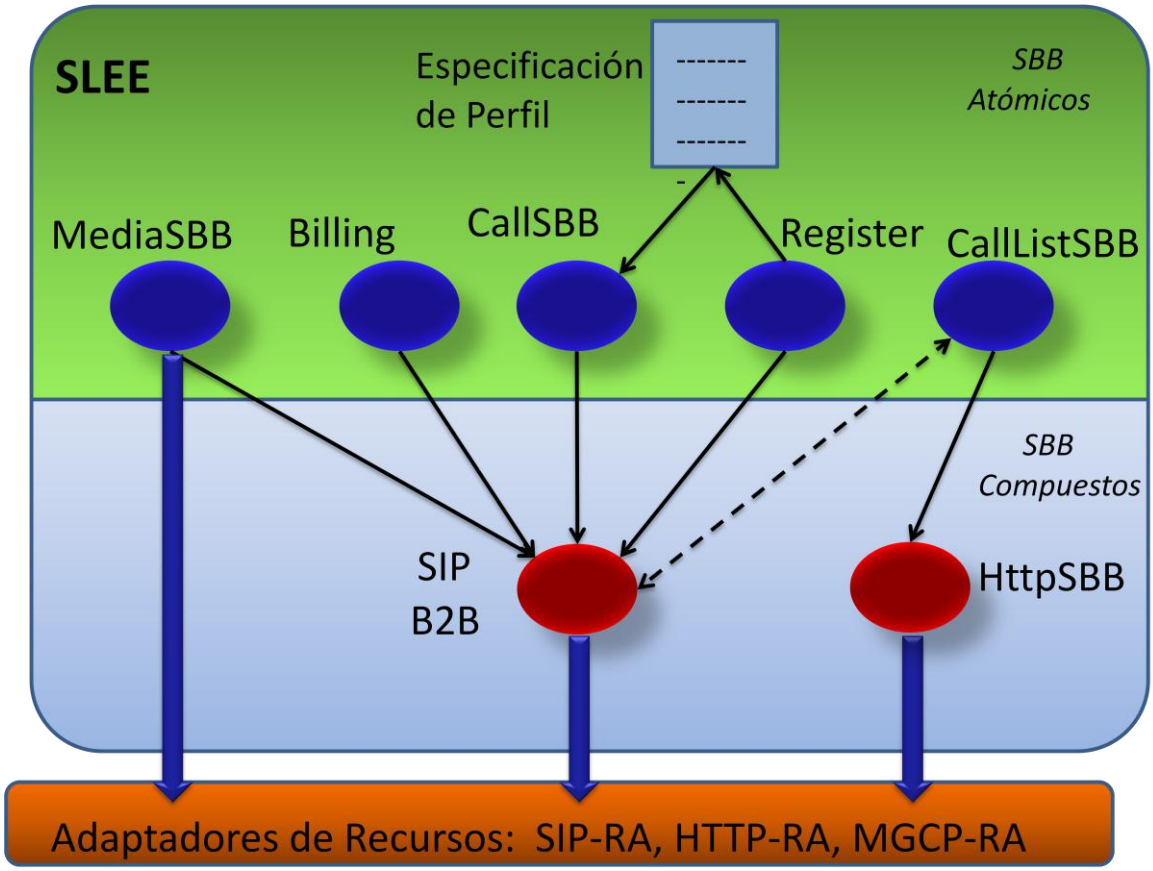

Figura 38. *Clasificación de SBB del caso de uso: Llamar Clientes (propia).*

En la figura 38 se muestra los SBB identificados con base en los sub-procesos definidos anteriormente. Como ya fue mencionado se reutilizan los SBB del caso de uso Llamada Ubicua y además se agregan tres SBB con la siguiente funcionalidad: el *HttpSBB* se encarga de iniciar el proceso de llamado automático una vez reciba la petición HTTP proveniente de la aplicación Web; el *CallListSBB* se encarga de gestionar el ciclo de las llamadas enviando peticiones al SBB *SIP B2B*, el cual realiza las llamadas; el *MediaSBB* es el encargado de interactuar con el servidor de medios para reproducir el archivo de audio. Finalmente es necesario resaltar de la grafica 38 las líneas punteadas entre algunos SBB, las cuales representan una composición de tipo asíncrona mientras que las líneas continuas una composición de tipo síncrona. Estas relaciones de composición serán explicadas con más detalle posteriormente.

5.2.2.4 Diagramas de Secuencia

En esta sección se presentan los diagramas de secuencia de los casos de uso del negocio, los cuales fueron realizados con base en los sub-procesos identificados anteriormente, pero teniendo en cuenta los SBB y las relaciones definidas para cada caso de uso.

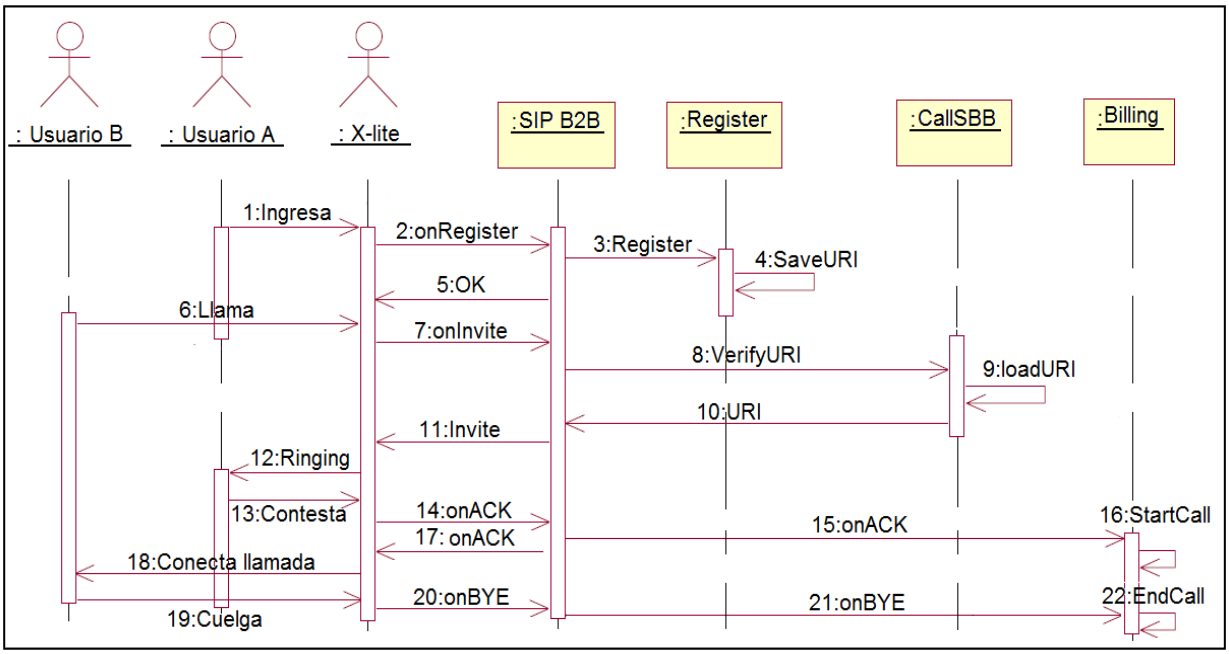

Figura 39. *Diagrama de Secuencia del caso de uso: Llamada Ubicua (propia).*

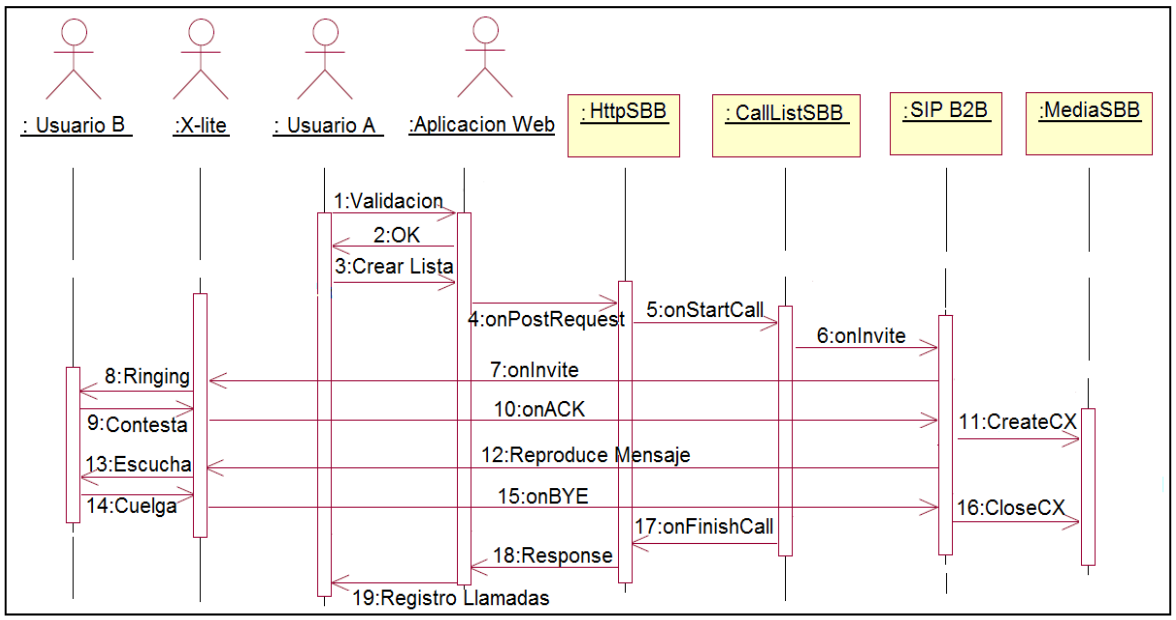

Figura 40. *Diagrama de Secuencia del caso de uso: Llamar Clientes (propia).*

En la figura 39 y 40 es posible observar la secuencia de eventos de cada caso de uso desde el punto de vista de sus SBB. Vale la pena resaltar que el SBB *SIP B2B* es reutilizado en el caso de uso Llamar Clientes, por lo tanto en la figura y este SBB es tomado como si fuera un SBB atómico, pero realmente se comporta de la misma forma que se describe en la figura 39.

Los eventos o métodos definidos en los anteriores diagramas de secuencia serán explicados con mayor detalle en la siguiente sección.

5.2.3 Especificación de SBB

Como se menciono en los lineamientos del Capítulo IV, en la especificación de SBB serán descritos detalladamente todos los SBB definidos en la identificación de bloques constructores del servicio. Para esta descripción se utilizara la plantilla propuesta en la sección 4.2.3 del Capítulo IV, tomando inicialmente los SBB del caso de uso: Llamada Ubicua, y posteriormente los del caso de uso: Llamar Clientes.

Caso de Uso: Llamada Ubicua

Característica	Descripción
Nombre de SBB	Register
Proveedor	Unicauca
Versión	1.0
Descripción	Este SBB se encarga de obtener y almacenar en la tabla de una especificación de perfil, la URI SIP, el número principal y el número alterno, además guarda un indicador que es utilizado para determinar en qué numero se encuentra disponible un usuario.
Nombre clase abstracta	RegistrarSbb

Nombre interfaz local	RegistrarSbbLocalObject
Nombre especificación de perfil	BPortableProfileCMP
Campos CMP	La especificación de perfil contiene cuatro campos CMP: • Número: Identifica a cada usuario del sistema. • URI: utilizada para ubicar al usuario dentro de la red. • NúmeroP: Es el numero alterno del usuario. • bandera B: Toma el valor de 1 o 0 dependiendo del número en que está disponible el usuario.
SBB hijos	Este es un SBB atómico, por lo tanto no tiene SBB hijos.
Eventos	Este SBB no envía o recibe ningún evento.
Métodos	Posee 3 métodos: • CreateProfileTable: Crea una tabla de perfiles en caso de que aun no haya sido creada. • CreateProfile: Crea el perfil del usuario con los campos CMP mencionados. • Register: Se encarga de editar el perfil del usuario de modo que la bandera B indique en que numero está disponible el usuario.
Adaptadores de recursos	Este SBB no utiliza ningún adaptador de recursos.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 5: *Especificación del SBB: Register (propia).*

Característica	Descripción
Nombre de SBB	CallSBB
Proveedor	Unicauca
Versión	1.0
Descripción	Este SBB se encarga de extraer de la tabla de una especificación de perfil, la información necesaria que le permita saber el número en el cual un usuario está disponible.
Nombre clase abstracta	CallSbb
Nombre interfaz local	CallSbbLocalObject
Nombre especificación de perfil	BPortableProfileCMP
Campos CMP	La especificación de perfil contiene cuatro campos CMP: • Número: Identifica a cada usuario del sistema. • URI: utilizada para ubicar al usuario dentro de la red. • NúmeroP: Es el numero alterno del usuario. • bandera B: Toma el valor de 1 o 0 dependiendo del número en que está disponible el usuario.
SBB hijos	Este es un SBB atómico, por lo tanto no tiene SBB hijos.
Eventos	Este SBB no envía o recibe ningún evento.
Métodos	Posee 2 métodos: • VerifyName: Recibe como parámetro la URI

	SIP de un usuario, y se encarga de extraer el numero de dicho usuario. • VerifyURI: Busca en la tabla de una especificación de perfil la información de un usuario, y determina si el usuario está disponible en su número principal o su número alterno.
Adaptadores de recursos	Este SBB no utiliza ningún adaptador de recursos.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 6: *Especificación del SBB: CallSBB (propia).*

Característica	Descripción
Nombre de SBB	Billing
Proveedor	Unicauca
Versión	1.0
Descripción	El SBB Billing es el encargado crear un registro en una base de datos con la información de la llamada, tal como el origen, destino, fecha y duración.
Nombre clase abstracta	BillingSbb
Nombre interfaz local	BillingSbbLocalObject
Nombre especificación de perfil	Este SBB no utiliza ninguna especificación de perfil.
Campos CMP	No tiene campos CMP
SBB hijos	Este es un SBB atómico, por lo tanto no tiene SBB hijos.
Eventos	Este SBB no envía o recibe ningún evento.
Métodos	Posee 3 métodos: • ACK: Recibe como parámetro los datos de una llamada, y se encarga de almacenar en una base de datos el origen, destino, fecha y hora de inicio. • BYE: Toma los datos de la llamada y completa la información en el registro correspondiente a hora de finalización de la llamada y su duración • ServiceType: Almacenar en la base de datos el nombre del servicio que hace uso de este SBB.
Adaptadores de recursos	Este SBB no utiliza ningún adaptador de recursos.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 7: *Especificación del SBB: Billing (propia).*

Característica	Descripción
Nombre de SBB	SIPB2B
Proveedor	Unicauca
Versión	1.0
Descripción	El SBB SIP B2B es un SBB que permite conectar una llamada SIP entre los dos usuarios del sistema.
Nombre clase abstracta	B2BUASbb
Nombre interfaz local	No posee interfaz local.
Nombre especificación de perfil	Este SBB no utiliza ninguna especificación de perfil.
Campos CMP	Tiene 2 campos CMP (incomingDialog y outgoingDialog), que utiliza para almacenar las

	actividades de contexto correspondientes a cada conversación.
SBB hijos	Este SBB tiene 3 SBB hijos, los cuales como se menciono anteriormente son: Register, CallSBB y Billing.
Eventos	Este SBB tiene 5 eventos: • *onRegister:* Recibe una petición de registro e invoca al método Register del SBB *Register*, dándole como parámetro la URI SIP de usuario. • *onInvite:* Recibe una invitación de llamada e invoca al método VerifyURI del *CallSBB* para obtener el destino real, luego crea 2 diálogos SIP, uno entre el SLEE y el origen, y otro entre el SLEE y el destino, posteriormente envía la invitación de llamada al destino. • *onACK:* Indica que el destino ha contestado la llamada, luego conecta la llamada e inicia la conversación. En este evento se invoca el método ACK del SBB *Billing*. • *onBYE:* Indica la terminación de la llamada por parte de cualquiera de los dos usuarios. Además es invocado el método BYE del SBB *Billing*. • *onCancel:* Indica la cancelación de la llamada antes de ser establecida la conversación.
Métodos	Este SBB presenta 2 metodos: • forwardRequest: Redirige las peticiones SIP hacia su correcto destino. • forwardResponse: Redirige las respuestas SIP hacia su correcto destino.
Adaptadores de recursos	Este SBB utiliza un adaptador de recursos SIP versión 1.2. Este adaptador tiene un enlace llamado SipRA, con una interfaz de actividad de contexto llamada: slee/resources/jainsip/1.2/acifactory.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 8: *Especificación del SBB: SIPB2B (propia).*

Caso de Uso: Llamar Clientes

Característica	Descripción
Nombre de SBB	HttpSBB
Proveedor	Unicauca
Versión	1.0
Descripción	El SBB HttpSBB se encarga de iniciar el proceso de llamado automático una vez reciba la petición http proveniente de la aplicación Web.
Nombre clase abstracta	HttpSbb
Nombre interfaz local	No posee interfaz local.
Nombre especificación de perfil	Este SBB no utiliza ninguna especificación de perfil.
Campos CMP	No tiene campos CMP.
SBB hijos	Este SBB tiene como hijo al SBB *CallListSBB.*

Eventos	Este SBB tiene 3 eventos: • *onPostRequest:* Recibe una petición proveniente de una aplicación Web, que le indica que debe empezar a funcionar el servicio. Por lo tanto llama al método getCallList del SBB *CallListSBB*. • *onResponse:* envía este evento hacia la aplicación Web para indicarle que ya se realizo el proceso de llamado automático.
Métodos	No tiene métodos.
Adaptadores de recursos	Este SBB utiliza un adaptador de recursos HTTP versión 1.0. Este adaptador tiene un enlace llamado HttpServletRA, con una interfaz de actividad de contexto llamada: slee/resources/httpservlet/1.0.1/acifactory.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 9: *Especificación del SBB: HttpSBB (propia).*

Característica	Descripción
Nombre de SBB	CallListSBB
Proveedor	Unicauca
Versión	1.0
Descripción	El SBB CallListSBB se encarga de gestionar el ciclo de la lista de llamadas, enviando por cada llamada peticiones al SBB SIP B2B, el cual realiza las llamadas.
Nombre clase abstracta	CallListSbb
Nombre interfaz local	No posee interfaz local.
Nombre especificación de perfil	Este SBB no utiliza ninguna especificación de perfil.
Campos CMP	No tiene campos CMP.
SBB hijos	Este es un SBB atómico, por lo tanto no tiene SBB hijos.
Eventos	Este SBB tiene 2 eventos: • *fireInvite:* envía este evento al SIPB2B, indicándole el origen y destino de la llamada.
Métodos	Tiene un solo método llamado getListCall, que se encarga de buscar y traer de una base de datos la lista de llamadas requeridas por el usuario.
Adaptadores de recursos	Este SBB utiliza un adaptador de recursos SIP versión 1.2. Este adaptador tiene un enlace llamado SipRA, con una interfaz de actividad de contexto llamada: slee/resources/jainsip/1.2/acifactory. .
Alarmas	Este SBB no tiene alarmas programadas

Tabla 10: *Especificación del SBB: CallListSBB (propia).*

Característica	Descripción
Nombre de SBB	MediaSBB
Proveedor	Unicauca
Versión	1.0
Descripción	El MediaSBB es el encargado de interactuar con el servidor de medios para reproducir el archivo de audio

	cuanto se establezca una llamada.
Nombre clase abstracta	MediaSbb
Nombre interfaz local	MediaSbbLocalObject.
Nombre especificación de perfil	Este SBB no utiliza ninguna especificación de perfil.
Campos CMP	No tiene campos CMP.
SBB hijos	Este es un SBB atómico, por lo tanto no tiene SBB hijos.
Eventos	Este SBB tiene 3 eventos: • *onCreateConnection:* Envía este evento al servidor de medios para que establezca una conexión con la URI SIP que se le mande como parámetro. • *onNotificationRequestResponse:* Recibe este evento de confirmación, indicando que fue posible establecer la conexión. • *onDeleteConnctionResponse:* Recibe este evento cuanto termina la sesión de media y se cierra la conexión.
Métodos	Tiene 2 métodos: • createCX: Se encarga de crear un evento onCreateConnection con la URI SIP del usuario, y enviarlo al servidor de medios. • CloseCX: Es invocado por el SIPB2B en caso de que el usuario termine la llamada.
Adaptadores de recursos	Este SBB utiliza un adaptador de recursos MGCP versión 2.0. Este adaptador tiene un enlace llamado MGCP, con una interfaz de actividad de contexto llamada: slee/resources/jainmgcp/2.0/acifactory.
Alarmas	Este SBB no tiene alarmas programadas

Tabla 11: *Especificación del SBB: MediaSBB (propia).*

5.2.4 Composición de SBB

Como se definió en la sección 4.2.4 del Capítulo IV, existen 2 tipos de composición, la síncrona y la asíncrona, por lo tanto es necesario definir el tipo de composición que es utilizada entre los SBB de cada caso de uso del negocio.

Los SBB involucrados en el proceso de composición de cada uno de los servicios son identificados y descritos de forma de detallada en las secciones 5.2.2 y 5.2.3. Sin embargo vale la pena resaltar que dentro del proceso de desarrollo de cada uno de los SBB se reutilizaron los SBB *SIPB2B, Http_servletra* y *MGCP_callSBB,* a partir de los cuales se crearon los SBB *SIPB2B, HttpSBB* y *MediaSBB.* Los SBB restantes (*CallListSBB, Billing, Register y CallSBB*) fueron completamente desarrollados en el marco de este trabajo de grado. A continuación se describen los tipos de composición utilizados en cada uno de los casos de uso del negocio pertenecientes a los servicios *Llamada Recordatoria* y *Número Portable*.

En el caso de uso Llamada Ubicua perteneciente al servicio *Número Portable*, existe un SBB padre y tres SBB hijos, todas las relaciones son de tipo síncrono debido a que este servicio es bastante exigente en el sentido de que se espera obtener un gran número de llamadas simultaneas. Como dichas relaciones son síncronas, los tres SBB hijos no

envían o reciben ningún tipo de eventos, es utilizado el método de composición que obtiene la interfaz *SbblocalObject* del SBB descrita en la sección 4.2.4.1, e invoca los métodos a través de esta.

Para el caso de uso Llamar Clientes perteneciente al servicio *Llamada Recordatoria*, como lo muestra la figura 38, existen relaciones de los dos tipos, del tipo síncrona entre el SBB *SIPB2B* y sus SBB hijos *Register, CallSBB, MediaSBB* y *Billing*, y entre el SBB *HttpSBB* y su hijo *CallListSBB* (señaladas con líneas continuas). La relación de tipo asíncrona se presenta entre los SBB *CallListSBB* y *SIPB2B* (señalada con una línea punteada). Esta relacione de tipo asíncrono es implementada de esta forma dado que no se requiere un procesamiento exigente cuando se recupera la lista de llamadas y envía peticiones de llamadas al SIPB2B, además el bajo número de eventos que se manejan en esta parte de la lógica facilita la utilización de relaciones asíncronas.

5.2.5 Implementación de los Servicios

Una vez realizado el modelado del negocio, la identificación de los bloques constructores del servicio, especificación de los SBB y definidas las relaciones de composición de SBB, el siguiente paso el implementar todos componentes definidos teniendo en cuenta los pasos propuestos en la sección 4.2.5 del Capítulo IV. Los detalles y el código fuente de los servicios implementados se encuentran disponibles en el CD-ROM que cumple con las condiciones de entrega del presente trabajo de grado.

5.3 Funcionamiento de los Servicios

En esta sección se presentara paso a paso el funcionamiento de los servicios desarrollados, analizando las graficas que demuestran su real implementación dentro del ambiente del laboratorio.

Servicio: *Numero Portable* (caso de uso: Llamada Ubicua)

El servicio inicia cuando el usuario *David*, quien se encuentra subscrito al servicio de *Número Portable* ejecuta desde un computador el softphone X-lite con un número alterno llamado “david2” (figura 41).

Figura 41. *Número alterno de usuario David (propia).*

Cuando el usuario se registra como “david2”, en la tabla de la especificación de perfil es modificado el campo booleano llamado Portable a un valor True, el cual le indica al sistema que actualmente el usuario *David* se encuentra disponible en el numero alterno “david2”. Este cambio se muestra en la figura 42.

Name	Type	Access	Value	Description
Number	java.lang.String	RW	david	Attribute exposed for management
UriP	java.lang.String	RW	sip:david@192.175.16.116:5	Attribute exposed for management
Portable	boolean	RW	True False	Attribute exposed for management
NumberP	java.lang.String	RW	david2	Attribute exposed for management
ProfileDirty	boolean	R	False	Attribute exposed for management
ProfileWriteable	boolean	R	False	Attribute exposed for management

Figura 42. *Perfil de usuario David registrado como “david2” (propia).*

Posteriormente el usuario *Julián* también ingresa y se registra en el sistema con su número principal “julian”, como se observa en la figura 43.

Figura 43. *Número principal de usuario Julian (propia).*

Ahora el usuario *Julián* intenta llamar al usuario *David*, digitando su número principal: “david”, pero como el usuario *David* no se encuentra disponible en ese numero la llamada debería no ser contestada, sin embargo gracias al servicio de *Número Portable,* el sistema transfiere la llamada a su número alterno: “david2”. De esta manera al usuario *Julián* no le interesa saber en qué ubicación o número se encuentra disponible el usuario *David*, simplemente lo llama a su número principal.

Figura 44(a): *teléfono de usuario Julián* ***Figura 44(b):*** *teléfono de usuario David (propia).*

En la figura 44(a), se observa cuando el usuario *Julián* llama al usuario *David*, digitando su número principal: “david”. En la figura 44(b) se muestra cuando entra la llamada del usuario *Julián* al teléfono del usuario *David*, el cual fue registrado con su número alterno: “david2”.

Servicio: *Llamada Recordatoria* (caso de uso: Llamar Clientes)

Este servicio se inicializa a través de una interfaz Web donde un usuario se autentica y posteriormente ingresa la lista de números a los cuales desea que el sistema llame (figuras 45 y 46).

Figura 45. *Autenticación de usuarios (propia).*

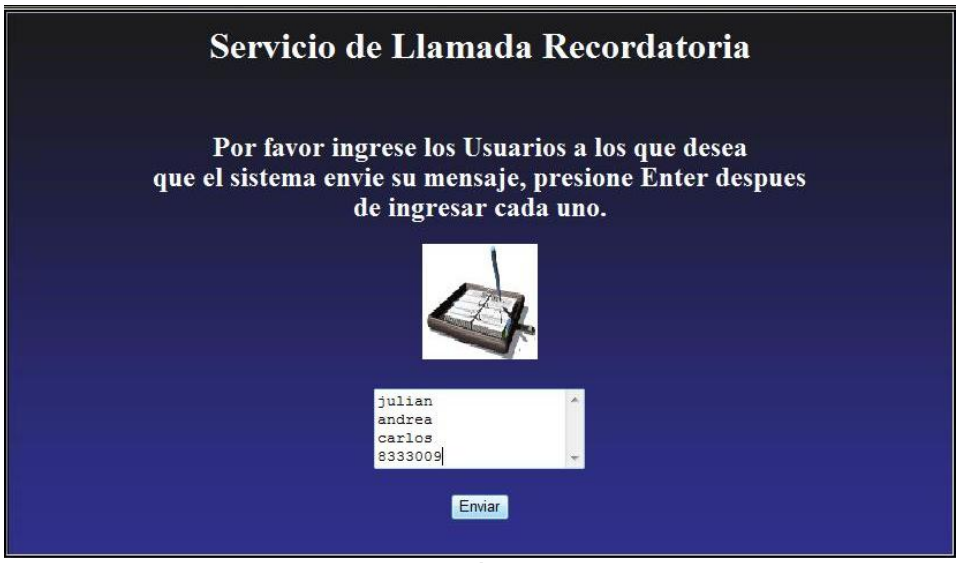

Figura 46. *Ingreso de números a llamar (propia).*

El sistema almacena la información en una base de datos y procede a realizar las llamadas indicadas en la lista provista por el usuario. Debido a que para el desarrollo de este servicio se reutilizaron los componentes desarrollados en el servicio *Número Portable,* el proceso de realización de las llamadas del servicio *Llamada Recordatoria* corresponde al descrito anteriormente para el servicio *Número Portable,* diferenciándose en que para *Llamada Recordatoria* las llamadas son originadas desde el SLEE y no por un usuario externo. En la figura 47 se observa una llamada entrante, originada desde el SLEE.

Figura 47. *Llamada originada desde el SLEE (propia).*

Una vez contestada la llamada originada por el servicio, se reproduce un mensaje predeterminado. Finalmente, a través de la interfaz Web el usuario tiene la posibilidad de

ver un registro que indica si las llamadas programadas por el usuario fueron exitosas o no, como se observa en la figura 48.

Figura 48. *Registro de Llamadas (propia).*

5.4 Resultados de los Servicios

Para evaluar los resultados obtenidos con el desarrollo de los servicios, es necesario analizar cada servicio desde dos puntos de vista muy importantes, el primero es el rendimiento de cada servicio respecto a los valores de desempeño que manejan los operadores de telecomunicaciones, y el segundo es el tiempo que tardó cada servicio en ser desarrollado, es decir el ya mencionado *Time-To-Market*.

Tiempos de desarrollo:

Antes de consignar los valores de los tiempos de desarrollo empleados para cada servicio, se debe definir una referencia del *Time-To-Market* con el fin de de terminar si los tiempos empleados son aceptables o no. La referencia para realizar una comparación la encontramos en [66], articulo en el cual las compañías Nokia Siemens Networks e IBM, hablan de su nueva plataforma para despliegue de servicios (SDP), la cual permite disminuir en tiempo desarrollo de servicios en un 80%, pasando de tardar 6 meses a tardar dos semanas y media para desarrollar nuevos servicios. Dicha plataforma es basada en los principios de reutilización de SOA e incluye un entorno de desarrollo gráfico bastante robusto, sin embargo cabe resaltar que esta plataforma es de tipo propietario y por lo tanto su adquisición tiene un elevado costo.

Teniendo en cuenta dicha referencia de tiempos de desarrollo de servicios, a continuación se realiza una breve comparación con el tiempo empleado para el desarrollo de cada uno de los servicios realizados en el marco de este trabajo de grado.

Nombre del Servicio	**Tiempo Empleados**	**Referencia [66]**
Numero Portable	16 días	18 días
Llamada Recordatoria	6 días	18 días

Tabla 12: *Tiempos de desarrollo de servicios (propia).*

En la tabla 12 se observa la comparación entre los tiempos de desarrollo empleados. Analizando estos valores concluimos que los servicios desarrollados dentro de este proyecto de grado, tiene tiempos aceptables, reflejándose de esta manera la valides del desarrollo de servicios basado en los lineamientos para composición de servicios de telecomunicaciones planteados en marco de este proyecto.

Otro aspecto que es necesario resaltar es el hecho de que los lineamientos planteados dentro de este trabajo de grado se basan en la reutilización de componentes, factor que disminuye aun más el *Time-To-Market*. Para evidenciar el efecto de dicha reutilización se comparan los tiempos empleados en el desarrollo de los 2 servicios, *Número Portable* y *Llamada Recordatoria*, siendo menor en un 64% el tiempo de desarrollo del servicio *Llamada Recordatoria*. Esta disminución de tiempo se obtuvo debido a que para desarrollar este servicio se llevó a cabo la reutilización en su totalidad de los componentes del servicio *Número Portable.*

Pruebas de desempeño:

Para la realización de las pruebas de desempeño de los servicios *Número Portable* y *Llamada Recordatoria*, se tomó como punto común entre los dos servicios el proceso de establecimiento y mantenimiento de una llamada, debido a que este proceso representa el punto principal en los dos servicios y es el que comparten a nivel de su estructura lógica.

El proceso de establecimiento y mantenimiento de una llamada para los dos servicios consiste en recibir el mensaje SIP *Invite,* verificar la existencia del usuario y sus datos de contacto en la especificación de perfil y realizar el intercambio de mensajes hasta el establecimiento de la llamada. En el servicio *Número Portable* el mensaje SIP *Invite* es originado desde un cliente SIP externo, mientras que para el servicio *Llamada Recordatoria* este mensaje es generado por el SBB *CallListSBB* al interior del SLEE. Una vez establecida la llamada se almacena en una base de datos la información de los usuarios y la hora de inicio de la llamada. Finalmente se recibe el mensaje *Bye* y se procede a terminar la llamada, almacenando en la base de datos la hora de su terminación y su duración.

Para esto se utilizó la herramienta SIPp [67]. Esta herramienta es un generador de tráfico para el protocolo SIP de libre distribución, a través de la cual es posible llevar a cabo las diferentes pruebas que permiten medir el desempeño de los servicios. Los parámetros utilizados para la medición del desempeño de los servicios fueron la cantidad de llamadas simultáneas que soportan y el número de llamadas por segundo que son capaces de procesar.

Los parámetros técnicos de las pruebas realizadas son los siguientes:

- Servidor JAIN SLEE Mobicents v1.2.7
- Ubuntu 8.04, Procesador Intel Core 2 Duo T7500 2,2GHz, 1GB RAM.
- SIPp v3.0

La forma utilizada para medir estos parámetros consistió en mantener un número límite fijo de llamadas simultáneas y variar el número de llamadas por segundo que establecían los servicios, observando la cantidad de llamadas que no se pudieron completar. De esta manera se pudo medir el número de llamadas por segundo que los servicios son capaces de procesar.

Para medir la cantidad de llamadas simultaneas que los servicios son capaces de mantener, se procedió a mantener un número fijo de llamadas por segundo y se observó cuantas llamadas simultáneas soportaban los servicios y el número de llamadas sin completar. A continuación se presentan los resultados de dichas pruebas.

En la figura 49 se observa que manteniendo un número de llamadas simultáneas fijo en 500 y a una tasa de 140 llamadas por segundo se obtiene un número de llamadas erróneas de 3.

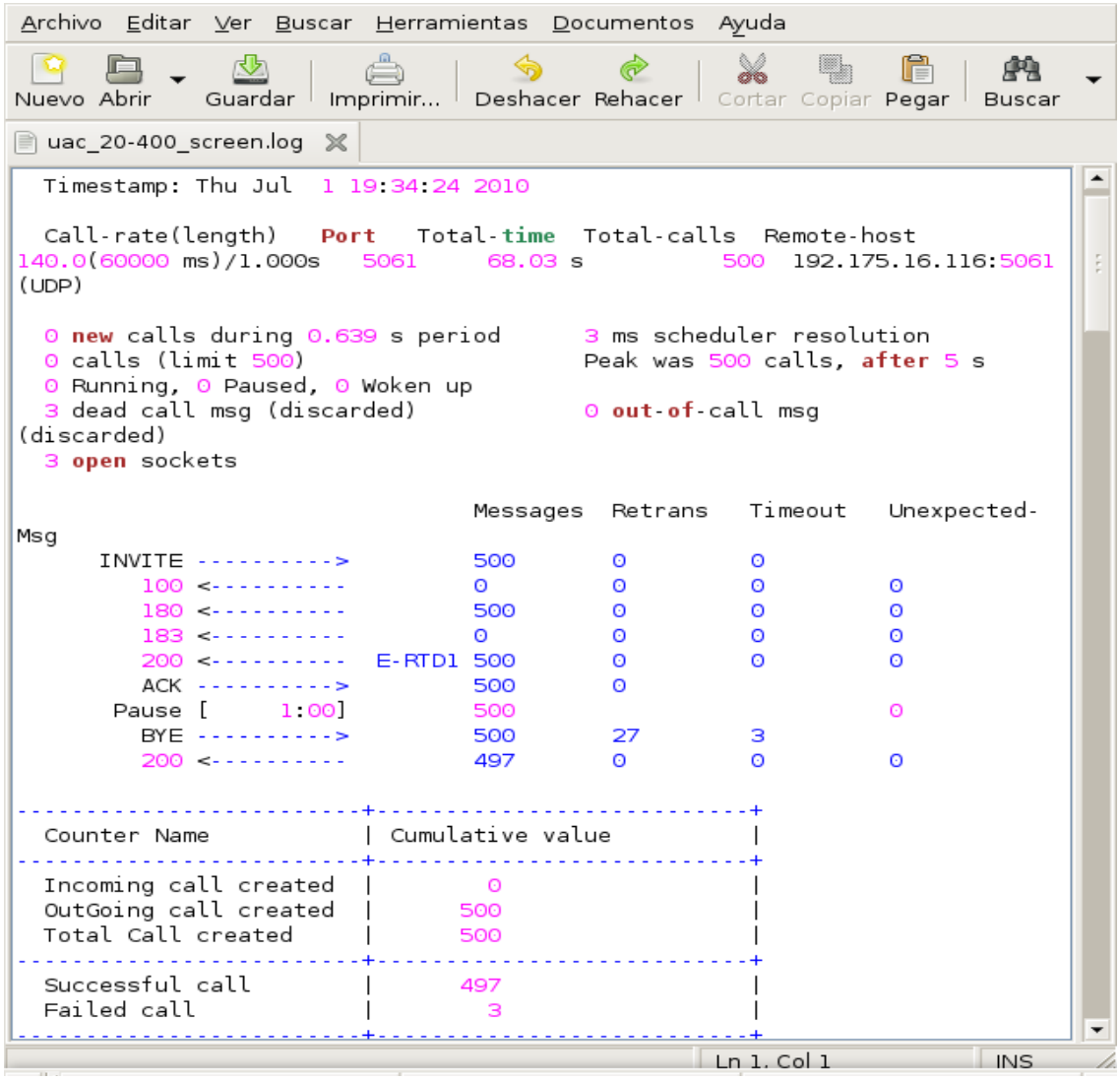

Figura 49. *Pruebas de desempeño: Número de llamadas por segundo (propia).*

De la misma manera, en la figura 50 se observa que manteniendo una tasa fija de 100 llamadas por segundo, se observa que de 900 llamadas no se completaron 19.

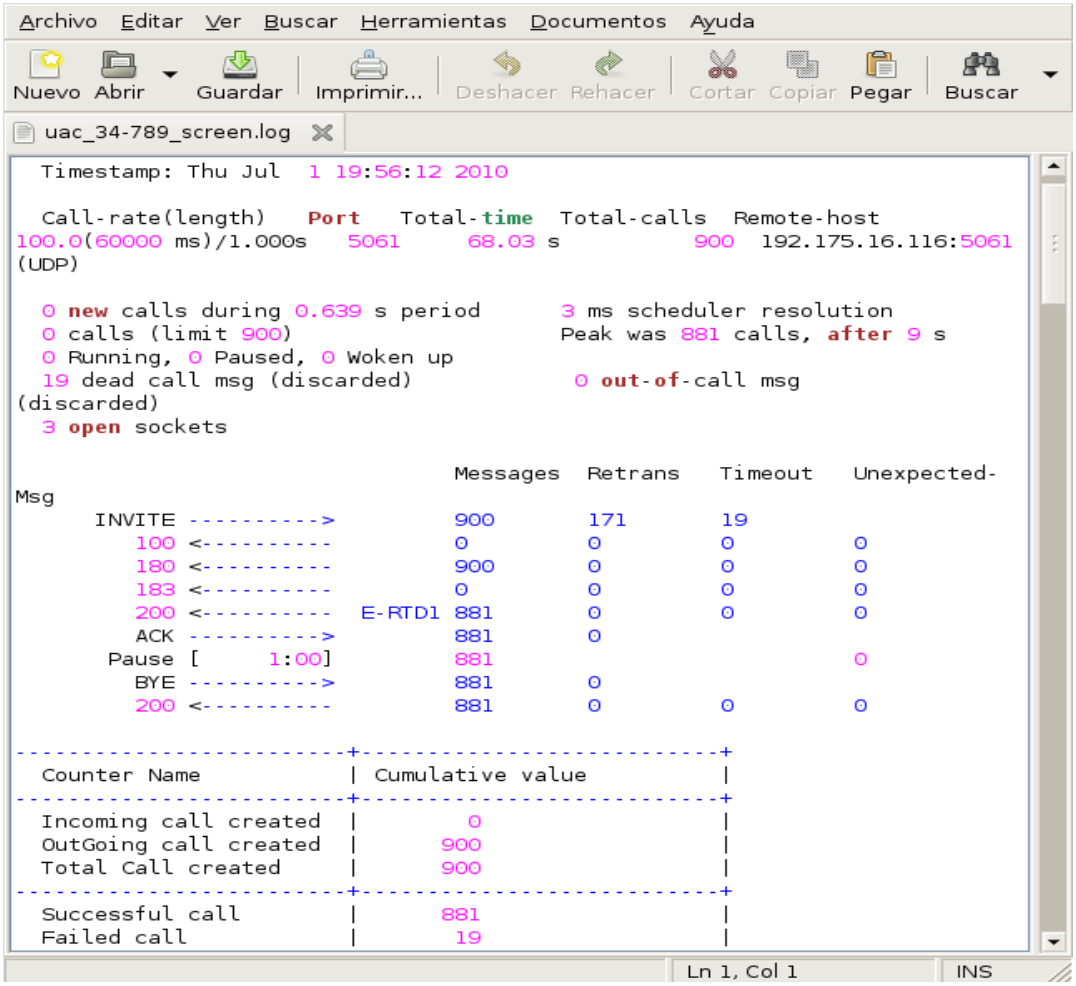

Figura 50. *Pruebas de desempeño: Número de llamadas simultáneas (propia).*

Por lo tanto, realizando variaciones a los parámetros anteriormente mencionados se obtienen las siguientes curvas.

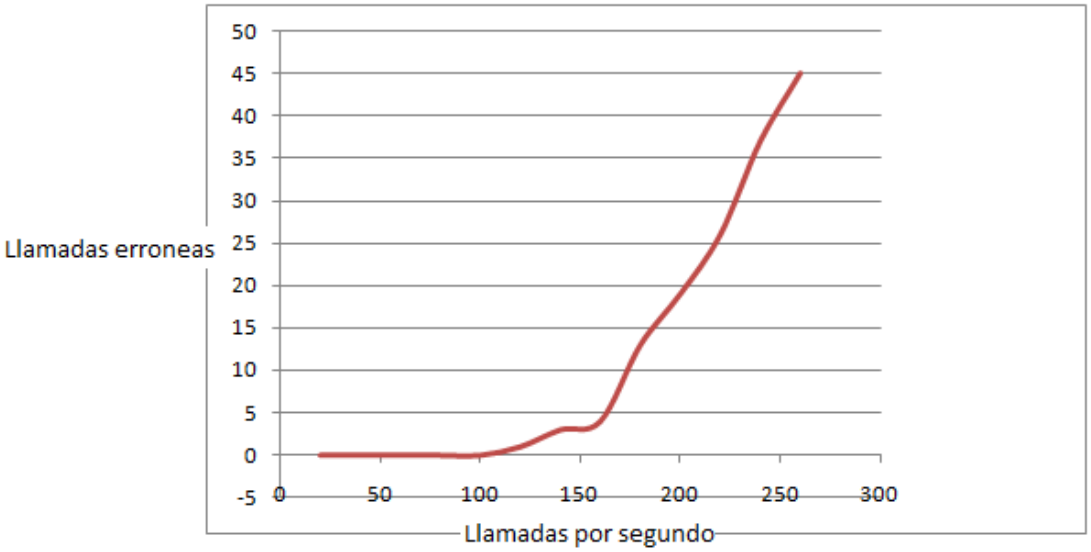

Figura 51. *Curva de Llamadas por segundo vs. Llamadas erróneas (propia).*

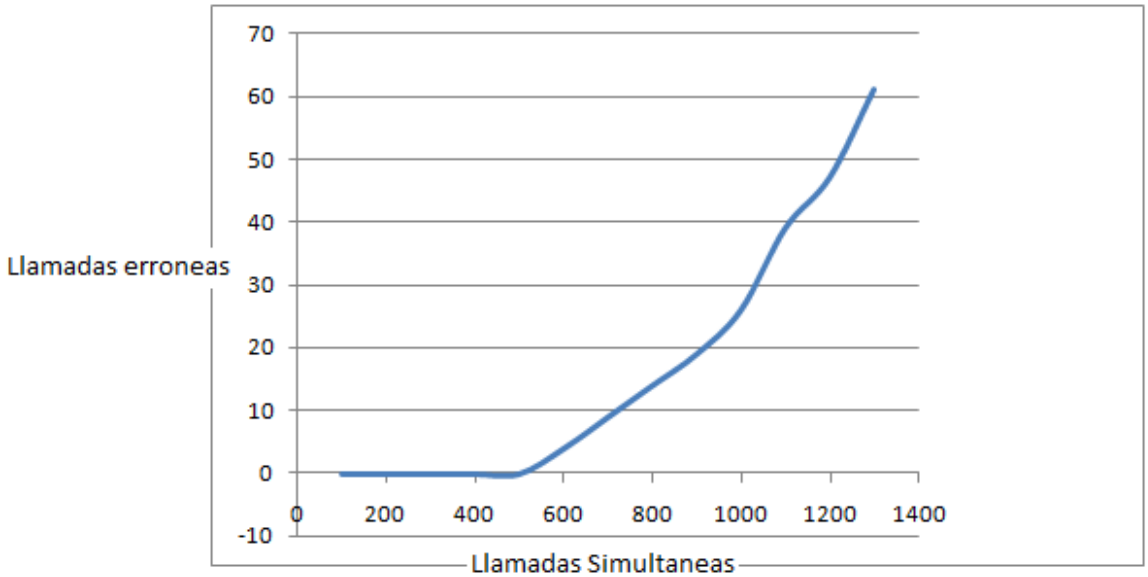

Figura 52. *Curva de Llamadas Simultáneas vs. Llamadas erróneas (propia).*

Como se observa en las figuras 51 y 52 , se tiene que el proceso de establecimiento y mantenimiento de llamada que presentan los servicios de *Número Portable* y *Llamada Recordatoria*, soportan una carga aproximada de establecimiento de 120 llamadas por segundo y 450 llamadas simultaneas sin presentar errores.

Estos resultados nos permiten concluir que para los dos servicios desarrollados en el marco de este trabajo de grado se obtienen resultados de desempeño aceptables que cumplen con los requisitos mínimos que presentan los servicios de telecomunicaciones de nivel de operador. Es posible realizar esta afirmación tomando en cuenta el análisis realizado en [68], donde se lleva a cabo un estudio de los requerimientos funcionales de los servicios VoIP de nivel de operador y se concluye que los servicios basados en el protocolo SIP deben ser capaces de procesar más de 90 llamadas por segundo. Como se mencionó anteriormente, los servicios *Número Portable* y *Llamada Recordatoria* tienen la capacidad de procesar aproximadamente 120 llamadas por segundo sin presentar errores, reflejando un resultado satisfactorio. En cuanto al número de llamadas simultáneas que son capaces de manejar los servicios, se toma como referencia a [69], donde en mediciones realizadas en un entorno de telecomunicaciones real se obtiene un número de llamadas simultáneas promedio de 1000. En comparación a los datos obtenidos en las pruebas de desempeño de los servicios *Número Portable* y *Llamada Recordatoria* se puede observar que los servicios no alcanzan a cumplir los niveles que demanda un entorno de telecomunicaciones real, sin embargo se debe tener en cuenta que el desempeño de los servicios está directamente relacionado con los equipos utilizados en su ejecución. Por lo tanto utilizando equipos de con mayor capacidad de procesamiento sería posible satisfacer dichos requerimientos.

RESUMEN

En este capítulo se presento la validación de los lineamientos para composición de servicios de telecomunicaciones JAIN SLEE propuestos en el Capítulo IV, siguiendo paso a paso los lineamientos y utilizando el entorno de desarrollo definió en el Capítulo III. Inicialmente se realizo el modelado del negocio, luego la identificación y especificación de los servicios y finalmente su implementación. Posteriormente en este capítulo fue presentado el funcionamiento de los servicios de una forma grafica que evidencio su real funcionamiento dentro del laboratorio. Para finalizar este capítulo se realizo un análisis de resultados teniendo en cuenta el desempeño del los servicios con el número de llamadas simultaneas, y los tiempos empleados en su desarrollo, para lo cual se tomo como referencia los tiempos promedio propuestos por la compañía Nokia Siemens Networks.

Capítulo VI

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo serán presentadas las conclusiones a las cuales se llego con el desarrollo de este trabajo de grado, además se darán algunas ideas sobre trabajos futuros que se podrían realizar teniendo como base este trabajo.

6.1 Aportes

- Se generó una base conceptual sobre el desarrollo de servicios en un entorno JAIN SLEE, resaltando las ventajas y capacidades que tiene esta especificación sobre la forma de crear nuevos servicios de telecomunicaciones por parte de los operadores y proveedores de servicios.

- El planteamiento del entorno compuesto por los Plug-in de Alcatel-Lucent y EclipSLEE, permitió ofrecer a la comunidad del software libre, una nueva alternativa para el desarrollo de servicios JAIN SLEE, brindándoles un entorno robusto y completo, el cual permite crear servicios convergentes mediante la integración de una variedad de protocolos y tipos de redes de acceso. Además esta alternativa es de gran interés para la PYMES, quienes no tienen los recursos suficientes para adquirir costosos entornos propietarios.

- Se plantearon una serie de lineamientos o pautas a seguir para desarrollar de una forma fácil y rápida nuevos servicios convergentes de telecomunicaciones. Para el planteamiento de estos lineamientos se tuvo como referencia los principios y conceptos definidos por SOA, tal como la reutilización y la escalabilidad. Los lineamientos planteados guían al desarrollador desde la identificación de un servicio a partir de una idea de negocio, hasta obtener implementados robustos servicios.

- Se realizó un aporte a la comunidad de software libre debido a la definición de lineamientos para composición de servicios JAIN SLEE, enteramente basado en herramientas y entornos de ejecución de libre distribución.

6.2 Conclusiones

- Fue realizado un análisis de los trabajos relacionados con el desarrollo y composición de servicios de telecomunicaciones, en el cual se observo la carencia de metodologías o pautas que permitieran llevar a cabo estos procesos de una manera más fácil y rápida.

- Se realizó una evaluación completa y detallada de los 3 entornos de desarrollo y composición de servicios JAIN SLEE de libre distribución, Alcatel-Lucent JAIN SLEE Plug-in, EclipSLEE y OpenCloud JAIN SLEE Plug-in. En esta evaluación se identificaron las ventajas y desventajas de cada entorno, finalmente se tomo la decisión de utilizar los entornos EclipSLEE y Alcatel-Lucent de forma conjunta,

obteniendo de esta manera un entorno más completo y con mejores características que cada uno de los entornos tomados por separado.

- Los lineamientos planteados fueron validados mediante el desarrollo de dos servicios, el servicio *Número Portable* y el servicio *Llamada Recordatoria*. Estos servicios se desarrollaron siguiendo estrictamente cada lineamiento y además utilizando el entorno de desarrollo propuesto en el marco de este trabajo de grado.

- Se realizó un análisis de resultados a los servicios desarrollados en este trabajo de grado, teniendo en cuenta factores como el *Time-to-Market* de cada servicio, el cual tuvo un valor de 16 días sin reutilizar componentes, y 6 días reutilizando el 50% de los componentes del servicio. Este valor es bastante bueno teniendo como referencia el *Time-to-Market* promedio (18 días) definido por Nokia Siemens Networks [66].

- Fueron tomadas algunas medidas del desempeño de los servicios, las cuales corresponden a los valores de la cantidad máxima de llamadas simultáneas que soporta el SLEE, y la cantidad máxima de llamadas por segundo que puede establecer. Se obtuvieron los valores de 450 llamadas simultáneas y 120 llamadas por segundo sin presentar errores, dichos valores están dentro del umbral exigido por los operadores de telecomunicaciones [68-69].

- Finalmente se concluye que este trabajo ofrece una nueva alternativa, que permite desarrollar servicios convergentes de telecomunicaciones, de una forma fácil, rápida y organizada.

6.3 Trabajo Futuro

En esta sección se plantean 3 ideas de trabajos futuros, las cuales tienen como objetivo la continuación de este trabajo de grado.

- Adaptar el modelo de SOA, es decir definir una arquitectura que contenga un repositorio de SBB, los cuales puedan ser accedidos mediante descriptores XML estándares. De esta manera se podría implementar un ESB para gestionar el acceso y comunicación de los SBB. Además con este modelo se podrían implementar métodos de búsqueda semánticos para recuperar SBB según el contexto y preferencias del usuario.

- Mejorar el entorno de composición de servicios, implementando una serie interfaces graficas que permitan al usuario realizar la composición mediante una barra de herramientas. De esta forma el desarrollador de servicios evitaría realizar la composición por medio de la escritura de código de programación puro.

- Integrar el SLEE dentro de una arquitectura IMS, debido a que actualmente muchos operadores de telecomunicaciones están migrando a este tipo de redes IP. Por lo tanto con IMS se tendría una arquitectura robusta que controle el acceso de red y el perfil de los usuarios, los cuales accederían a los servicios desarrollados con los lineamientos propuestos en este trabajo de grado.

- Implementar un entorno de desarrollo de servicios JAIN SLEE de libre distribución, el cual siga la especificación JAIN SLEE versión 1.1. Este trabajo es planteado debido a que los tres entornos de libre distribución evaluados en este trabajo de grado, siguen la especificación JAIN SLEE versión 1.0.

REFERENCIAS

[1] Telco 2.0 Initiative, 2010, http://www.stlpartners.com/telco2.php.

[2] N. Pérez, H. Muñoz, D. Marcos, J. Martínez, Gestión de Procesos de Negocio Semánticos, Telefónica Investigación y Desarrollo, 2007.

[3] JSR 22: JAIN SLEE API, 2004, http://jcp.org/en/jsr/detail?id=22.

[4] JSR 240: JAIN SLEE v1.1, 2008, http://www.jcp.org/en/jsr/detail?id=240.

[5] Y. Yuan, J. Wen, B. Zhang, A Comparison of Three Programming Models for Telecom Service Composition, IBM China Research Laboratory, 2007.

[6] V. Ros, F. Fernandez, Aplicaciones de Telefonía sobre JAIN SLEE, Escuela Superior de Ingenieros, 2007.

[7] Mobicents Project, 2010, http://www.mobicents.org/.

[8] RFC 3261: Session Initiation Protocol, 2002, http://tools.ietf.org/html/rfc3261.

[9] RFC 2774: Hypertext Transfer Protocol, 2000, http://tools.ietf.org/html/rfc2774.

[10] RFC 3435: Media Gateway Control Protocol, 2003, http://tools.ietf.org/html/rfc3435.

[11] JSR 221: Java Database Connectivity API v4.0, 2005, http://jcp.org/aboutJava/communityprocess/edr/jsr221/index2.html.

[12] JSR 255: Java Management Extensions v2.0, 2008, http://www.jcp.org/en/jsr/detail?id=255.

[13] A. Johnston, J. Gabrielsson, Evolution of Service Delivery Platforms, Ericcson Review, 2007.

[14] Nokia Siemens Networks, Service Delivery Framework. 2008.

[15] J. Rojas, J. Ramírez, J. Corrales, Entornos de Desarrollo de Libre Distribución para Composición de Servicios JAIN SLEE, Universidad del Cauca, 2010.

[16] Quinto Seminario Nacional de Tecnologías Emergentes en Telecomunicaciones y Telemática, http://ieee.unicauca.edu.co/tet2010/.

[17] J. Montoya, E. Montoya, Servicios Convergentes en Redes de Próxima Generación, Universidad EAFIT, 2008.

[18] N. Silva, A. Shirato, F. Telbisz, S. Jakobsson, A. Ryan, S. Cherki, Service Oriented Architectures for Convergent Delivery Platforms, Portugal Telecom Inovação, NTT Japan, France Telecom Group, Telenor ASA, 2006.

[19] Microsoft Corporation, La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real, 2006.

[20] W3C Recommendation SOAP v1.2, 2007, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

[21] W3C Recommendation WSDL v2.0, 2007, http://www.w3.org/TR/wsdl20/.

[22] R. Kodali, What is Service-Oriented Architecture, 2005, http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html?page=1.

[23] A. Barco, Arquitectura Orientada a Servicios (SOA): Principios de la Orientación a Servicios, 2006, http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html.

[24] T. Erl, SOA Principles of Service Desing, Prentice Hall/PearsonPTR, ISBN 0132344823, 2007.

[25] OASIS UDDI Specification v3.0.2, 2004, http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm.

[26] Open Cloud Limited, A SLEE for all Seasons, 2007.

[27] Open Cloud Limited, An Introduction to JAIN SLEE, 2006.

[28] Open Mobile Alliance, 2010, http://www.openmobilealliance.org/.

[29] Telecoms and Internet converged Services and Protocols for Advanced Network, 2010, http://www.etsi.org/tispan/.

[30] 3rd Generation Partnership Project, 2010, http://www.3gpp.org/.

[31] World Wide Web Consortium, 2010, http://www.w3.org/.

[32] Internet Engineering Task Force, 2010, http://www.ietf.org/.

[33] Organization for the Advancement of Structured Information Standards, 2010, http://www.oasis-open.org.

[34] Open Cloud Limited, Rhino Overview and Concepts, 2009.

[35] Pendiente Ericsson SDP.

[36] D. Silas, Mobicents User Guide, Red Hat Documentation Group, 2007.

[37] C. Peltz, Web Services Orchestration: A Review of Emerging Technologies, Tools and Standards, Hewlett Packard Company, 2003.

[38] P. Falcarin, C. Venezia, Communication Web Services and JAIN SLEE Integration Challenges, Telecom Italia, Politecnico di Torino, 2008.

[39] The European IST-FP6 Project SPICE, 2008, http://www.ist-spice.org/index.html.

[40] Advanced Language for Value Added Services Composition and Creation, SPICE Consortium, 2006.

[41] F. Lécué, E. Silva, L. Ferreira, A Framework for Dynamic Web Service Composition, France Telecom R&D, University of Twente, The Netherlands, 2008.

[42] J. Martinez, Dynamic Service Composition in an Innovative Communication Environment, University of Twente, The Netherlands, 2008.

[43] A. Lehmann, R. Lasch, TeamCom: A Service Creation Platform for Next Generation Networks, University of Applied Sciences Frankfurt/Germany, 2009.

[44] T. Eichelmann, W. Fuhrmann, Creation of Value Added Services in NGN with BPEL, University of Applied Sciences Frankfurt/Germany, University of Plymouth United Kingdom, 2008.

[45] S. Bessler, J. Zeis, An Orchestrated Execution Environment for Hybrid Services, Kommunikation in Verteilten Systement, 2007.

[46] EclipSLEE Project, 2007, https://eclipslee.dev.java.net/.

[47] Alcatel-Lucent, Alcatel-Lucent SCE-SE Eclipse plug-in User Guide v1.3.10, 2008.

[48] Alcatel-Lucent, Open Service Platform (OSP) JAIN SLEE partner program, 2007.

[49] Open Cloud Limited, OpenCloud Eclipse Plug-in, 2008, https://developer.opencloud.com/devportal/display/OCDEV/Eclipse+Plugin+1.0.5.

[50] M. Rosen, B. Lublinsky, Applied SOA Service-Oriented Architecture and Desing Strategies, Wiley Publishing Inc, ISBN 978-0-470-22365-9, 2008.

[51] Alcatel-Lucent SCE-SE plug-in, https://sourceforge.net/projects/sce-se/.

[52] The JAIN SLEE Eclipse Plug-in: Gettiing Started, https://eclipslee.dev.java.net/html/GettingStarted.html.

[53] EclipSLEE Tutorial Flash Demo, http://groups.google.com/group/mobicents-public/web/AutomaticBuildAndDeployment.htm.

[54] EclipSLEE Home, https://eclipslee.dev.java.net/.

[55] EclipSLEE Tutorial Installation, http://groups.google.com/group/mobicents-public/web/eclipslee-tutorial?pli=1.

[56] Open Cloud JAIN SLEE Discussions, https://developer.opencloud.com/forum/forums/list.page.

[57] Eclipse Plug-in 1.0.5, https://developer.opencloud.com/devportal/display/OCDEV/Eclipse+Plugin+1.0.5.

[58] Unified Modeling Language, OMG, http://www.uml.org/.

[59] Rational Unified Process Whitepaper, IBM, http://www.augustana.ab.ca/~mohrj/courses/2000.winter/csc220/papers/rup_best_practices/rup_bestpractices.html.

[60] U. Wahil, L. Ackerman, A. Di Bari, Building SOA Solutions Using the Rational SDP, IBM Redbooks, 2007.

[61] N. Fareghzadeh, Service Identification Approach to SOA Development, World Academy of Science, Engineering and Technology, 2008.

[62] S. Inaganti, G. Krishna, Service Identification: BPM and SOA Handshake, BPTrends, 2007.

[63] Glassfish Community, https://glassfish.dev.java.net/.

[64] CounterPath Coporation X-Lite, http://www.counterpath.com/x-lite.html.

[65] PostgreSQL, http://www.postgresql.org/.

[66] Questex Media Group Company, http://www.enterpriseinnovation.net/content/philippine-telco-cuts-time-market-new-services-80.

[67] SIPp. http://sipp.sourceforge.net/.

[68] N. Schwan, T. Strauss, Peer-toPeer VoIP & MMoIP for Public Services – Requirements and Architecture, Alcatel-Lucent Deutschland, 2008.

[69] R. Birke, M. Mellia, M. Petracca, Understanding VoIP from Backbone Measurements, Politecnico di Torino, ENTS Telecom Paris, 2007.

ANEXO A: INSTALACIÓN DE HERRAMIENTAS

A continuación se presenta una descripción de los procesos de instalación de las herramientas necesarias para realizar desarrollo y composición de servicios JAIN SLEE en entornos de libre distribución. La instalación de las herramientas descritas a continuación es realizada en un sistema operativo Linux.

El proceso de instalación descrito en este anexo pertenece a las siguientes herramientas:

- Eclipse Europa v3.3.2
- Mobicents JAIN SLEE Server v1.2.7
- EclipSLEE v1.2.4
- Alcatel-Lucent (SCE-SE) v1.4.0
- X-Lite v2.0

Eclipse Europa v3.3.2:

Eclipse es el entorno de desarrollo en el cual es posible instalar los Plug-in que permiten la creación de servicios JAIN SLEE. Se recomienda la versión 3.3.2 debido a que existen incompatibilidades de los Plug-in con versiones posteriores de Eclipse. La versión 3.3.2 de Eclipse se puede descargar del siguiente enlace:

http://archive.eclipse.org/eclipse/downloads/drops/R-3.3.2-200802211800/index.php

Una vez descargado, se descomprime el archivo .zip y se ejecuta el archivo *eclipse* como se observa en la figura 53.

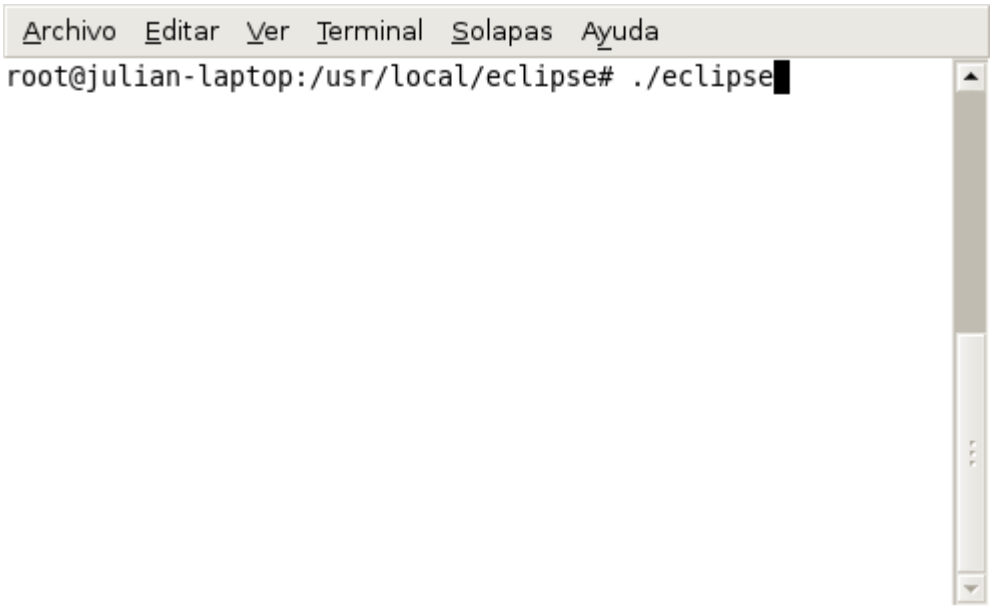

Figura 53. *Ejecución de Eclipse (propia).*

Mobicents JAIN SLEE Server v1.2.7:

Mobicents es el entorno de ejecución para servicios JAIN SLEE. Esta herramienta se puede descargar en el siguiente enlace:

http://hudson.jboss.org/hudson/view/Mobicents/job/MobicentsSlee1Release/

Una vez descargado, se debe descomprimir el archivo .zip. Posteriormente es necesario configurar la variable de entorno *JBOSS_HOME,* en el archivo *.bashrc,* indicando la ubicación de la carpeta donde fue descomprimido el archivo descargado, como se observa en las figuras 54 y 55.

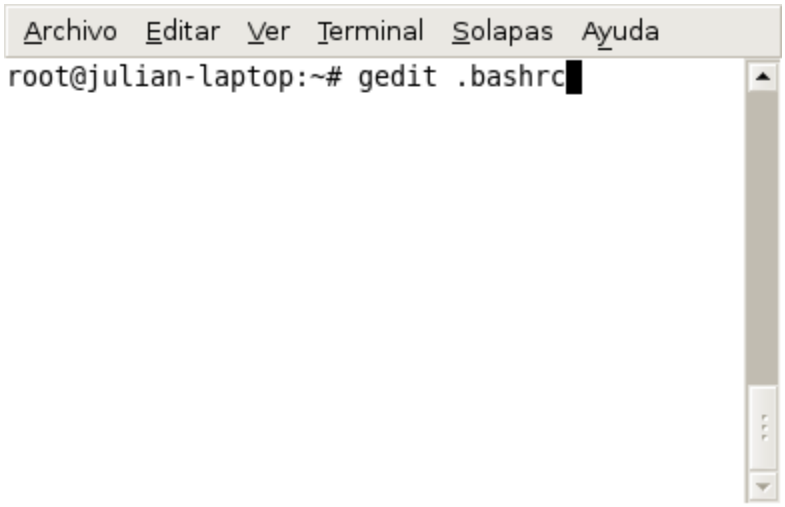

Figura 54. *Abrir archivo .bashrc (propia).*

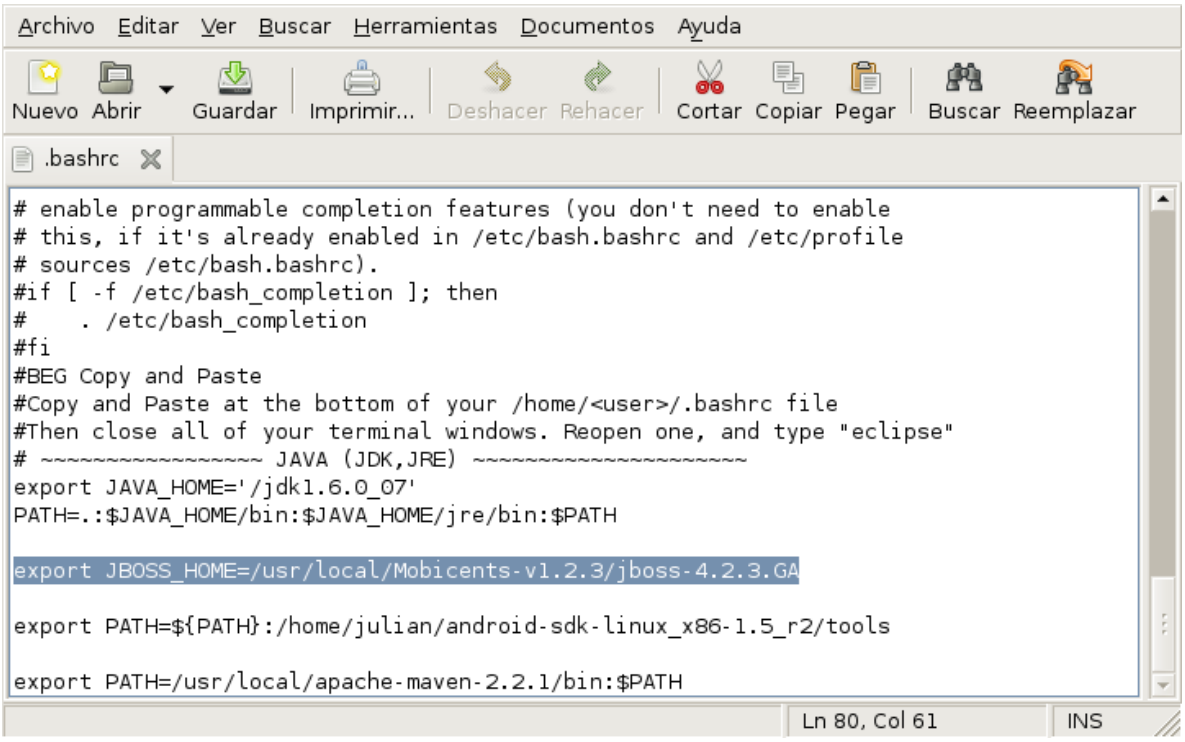

Figura 55. *Configuración de variable de entorno JBOSS_HOME (propia).*

Finalmente se ejecuta el archivo *run.sh* como se observa en la figura 56.

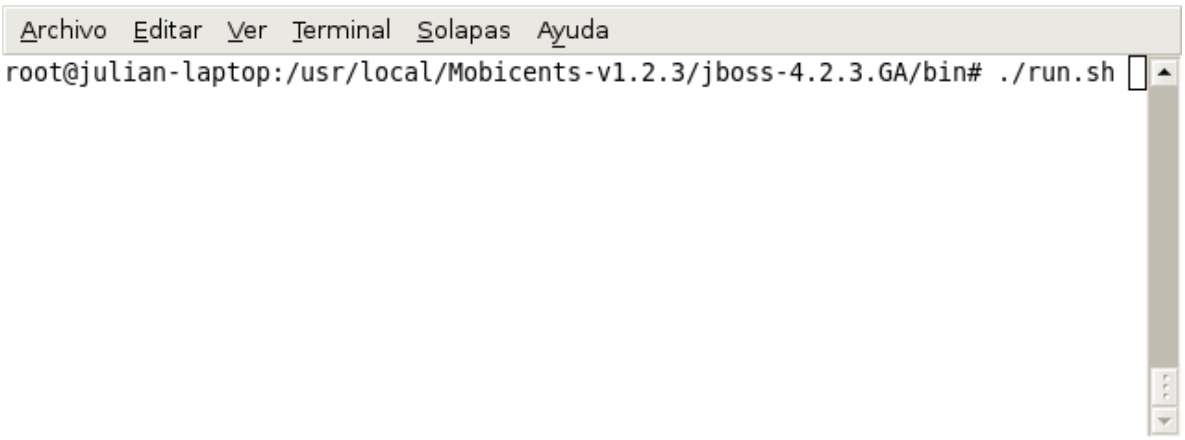

Figura 56. *Ejecución de Mobicents (propia).*

EclipSLEE v1.2.4:

EclipSLEE es una herramienta que permite crear componentes JAIN SLEE a través del entorno de desarrollo Eclipse. Se recomienda la versión 1.2.4 de EclipSLEE sobre Eclipse 3.3.2. Existen versiones posteriores de EclipSLEE que son soportadas en versiones posteriores de Eclipse, sin embargo estas versiones presentan errores. Para instalar EclipSLEE 1.2.4 se procede de la siguiente manera:

- Desde Eclipse se abre la opción *Help → Software Updates → Find and Install,* en la barra de herramientas superior.
- Se selecciona la opción *Search for new features to install*.
- Se selecciona la opción *New Remote Site* y se pasa como parámetro URL el siguiente enlace: http://people.redhat.com/vralev/eclipslee/update/ (figura 57).

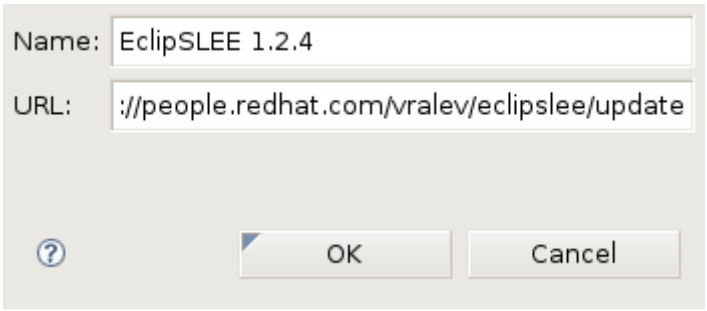

Figura 57. *Instalación de EclipSLEE(1) (propia).*

- Finalmente se selecciona el sitio recién creado y se da clic en la opción *Finish* (figura 58).

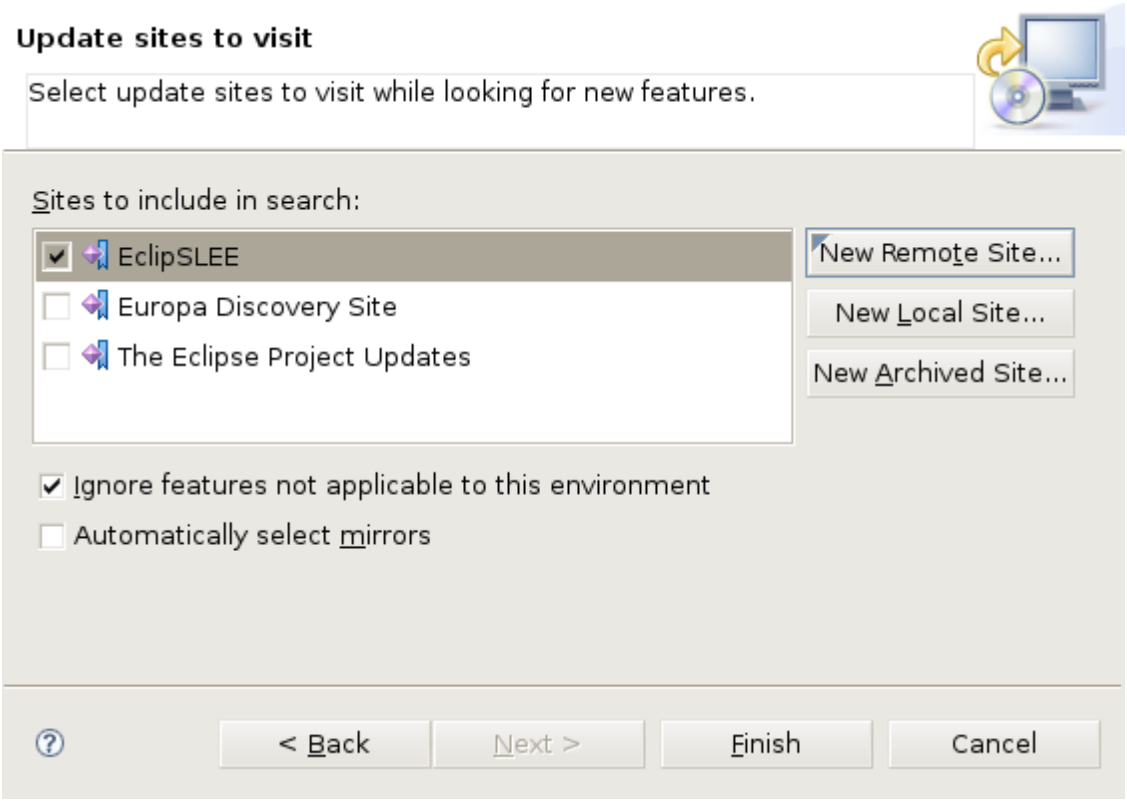

Figura 58. *Instalación de EclipSLEE(2) (propia).*

Alcatel-Lucent (SCE-SE) v1.4.0:

Alcatel-Lucent (SCE-SE) es una herramienta que permite crear componentes JAIN SLEE a través del entorno de desarrollo Eclipse. Para esta herramienta solo se encuentra disponible la versión 1.4.0 que funciona sobre Eclipse 3.3.2 en el siguiente enlace:

http://sourceforge.net/projects/sce-se/

El archivo descargado contiene las carpetas *features* y *plugins* (figura 59) cuyo contenido se debe copiar en las respectivas carpetas que contiene el directorio donde se encuentra instalado Eclipse (figura 60).

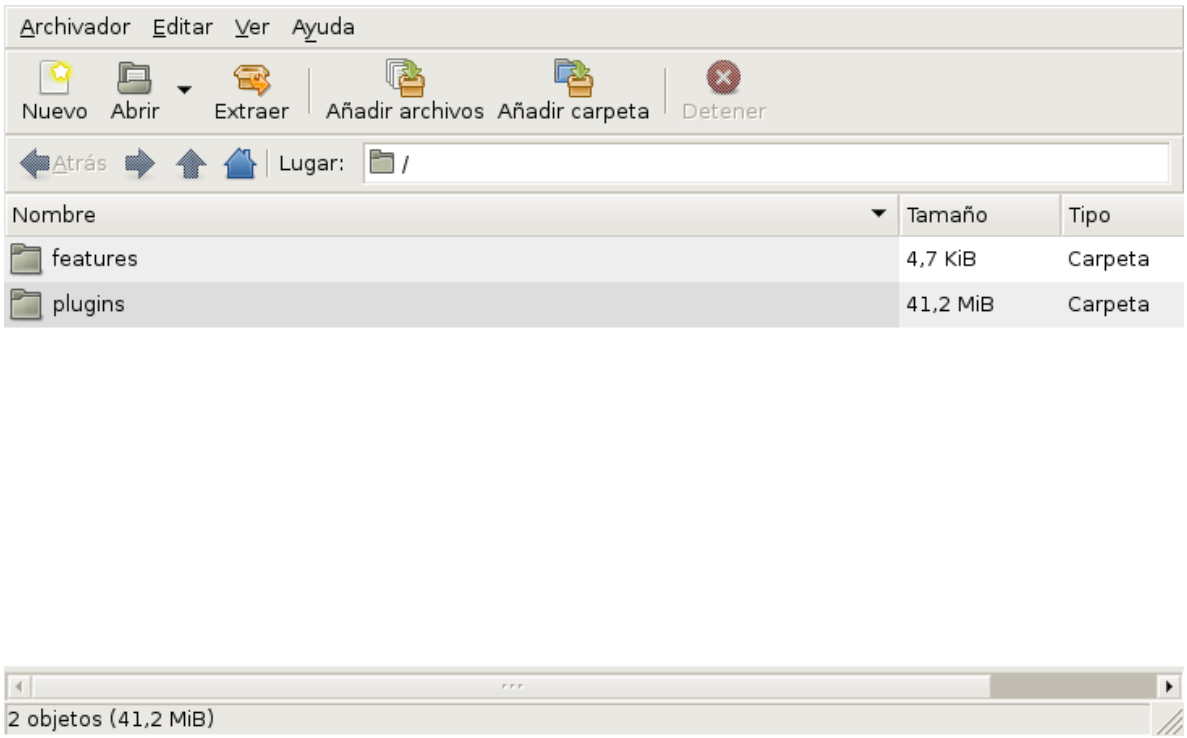

Figura 59. *Contenido del archivo de Alcatel-Lucent (SCE-SE) (propia).*

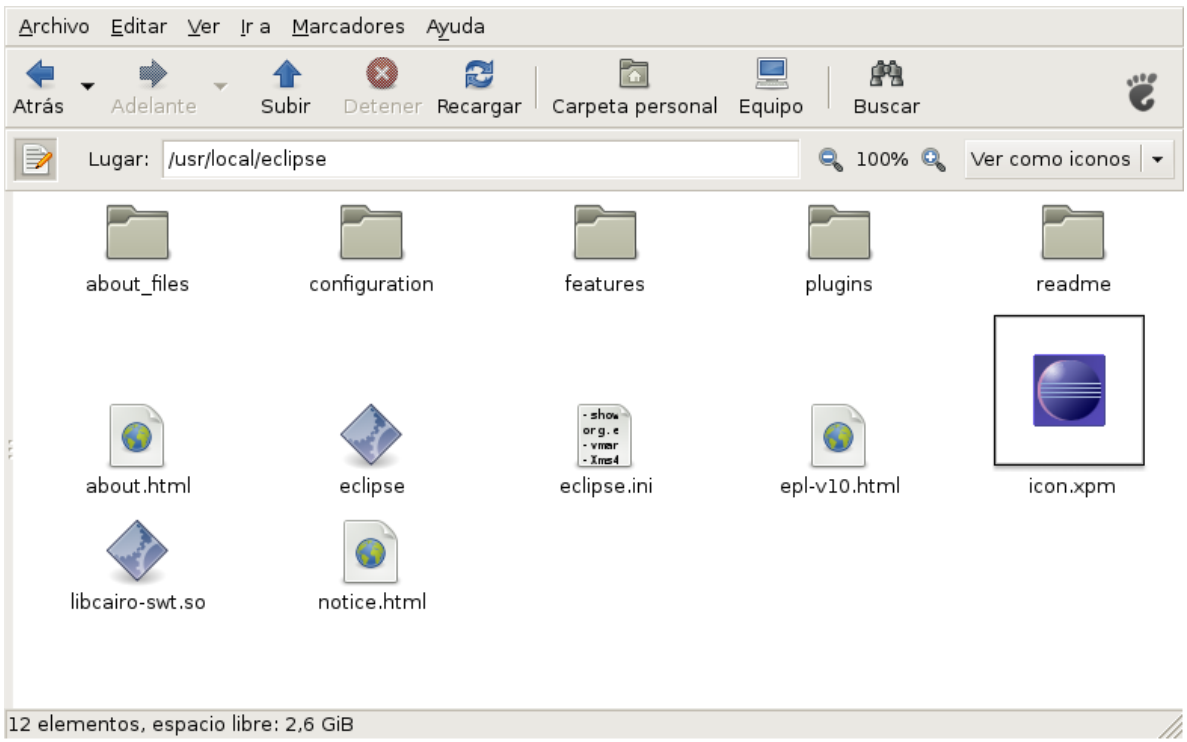

Figura 60. *Directorio de Instalación de Eclipse (propia).*

X-Lite v2.0:

X-Lite es un cliente SIP a través del cual se posible realizar registros y llamadas SIP utilizando los servicios JAIN SLEE. Esta herramienta se puede descargar en el siguiente enlace:

http://www.counterpath.com/x-lite-download.html

Una vez descargado, se descomprime el archivo con el comando *tar –xvfz* y se ejecuta como se observa en la figura 61.

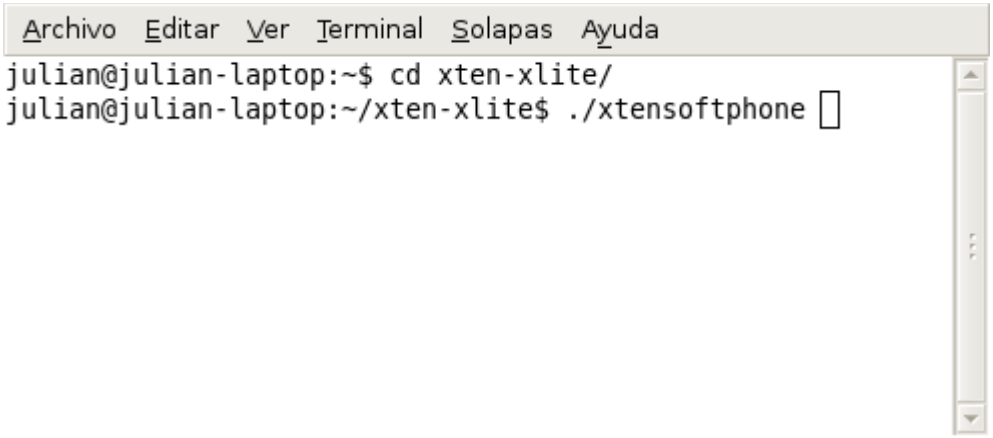

Figura 61. *Ejecución de X-Lite (propia).*

ANEXO B: INTEGRACIÓN DE ENTORNOS ECLIPSLEE Y ALCATEL-LUCENT

Como se menciono en el Capítulo III de la monografía, el mejor entorno para desarrollar servicios JAIN SLEE se obtiene mediante la integración de las herramientas Alcatel-Lucent SCE-SE y EclipSLEE, por lo tanto en este anexo será explicada paso a paso dicha integración.

Antes de empezar el proceso se debe descargar el archivo ECLIPSE_SCE_SE_1_4_0_SRC, del siguiente enlace:

http://sourceforge.net/projects/sce-se/

Posteriormente ya que tenemos instaladas estas herramientas como se explico en anexo A, ejecutamos Eclipse y procedemos a crear un nuevo proyecto de EclipSLEE como lo muestra la figura 26.

Luego debemos agregar la librería de Alcatel-Lucen al proyecto creado, para poder crear componentes de esta herramienta sobre nuestro proyecto de EclipSLEE, esto lo hacemos de la siguiente manera:

- Click sobre el proyecto → *Built Path* → *Add Library*

Debemos observar la figura 62 en nuestra pantalla.

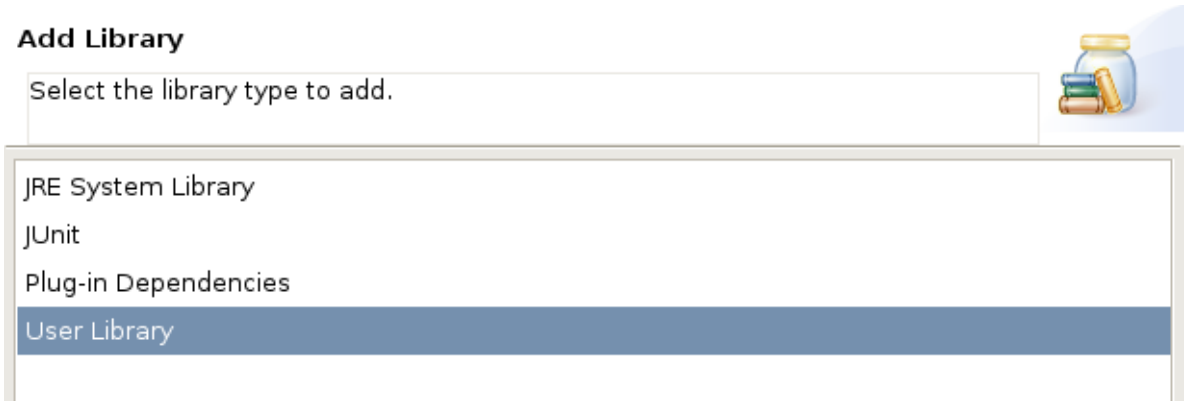

Figura 62. *Agregar una librería sobre un proyecto EclipSLEE (propia).*

Creo una nueva librería de usuario con el nombre AlcatelSCE.

- Click *User Library* → *New* → Escribo AlcatelSCE

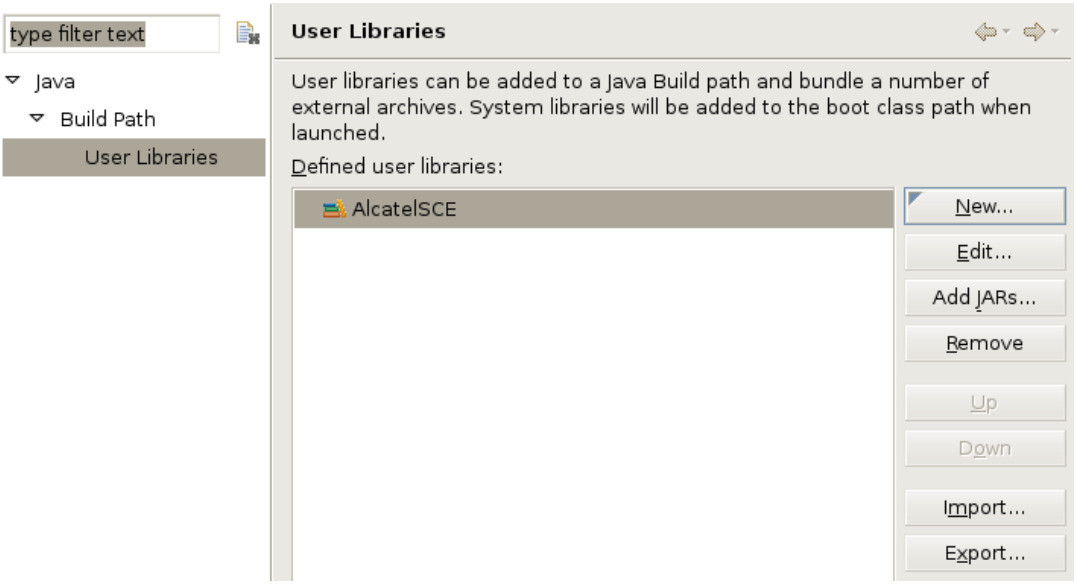

Figura 63. *Nueva librería de Alcatel SCE (propia).*

Agrego los archivos .jar a la librería creada.

- Click *AlcateSCE* → *Add JARs…* → *OK*

Añado todos los archivos que se encuentran en la carpeta descargada (figura 64):

ECLIPSE_SCE_SE_1_4_0_SRC [ubicación] / Eclipse-JSCE / lib

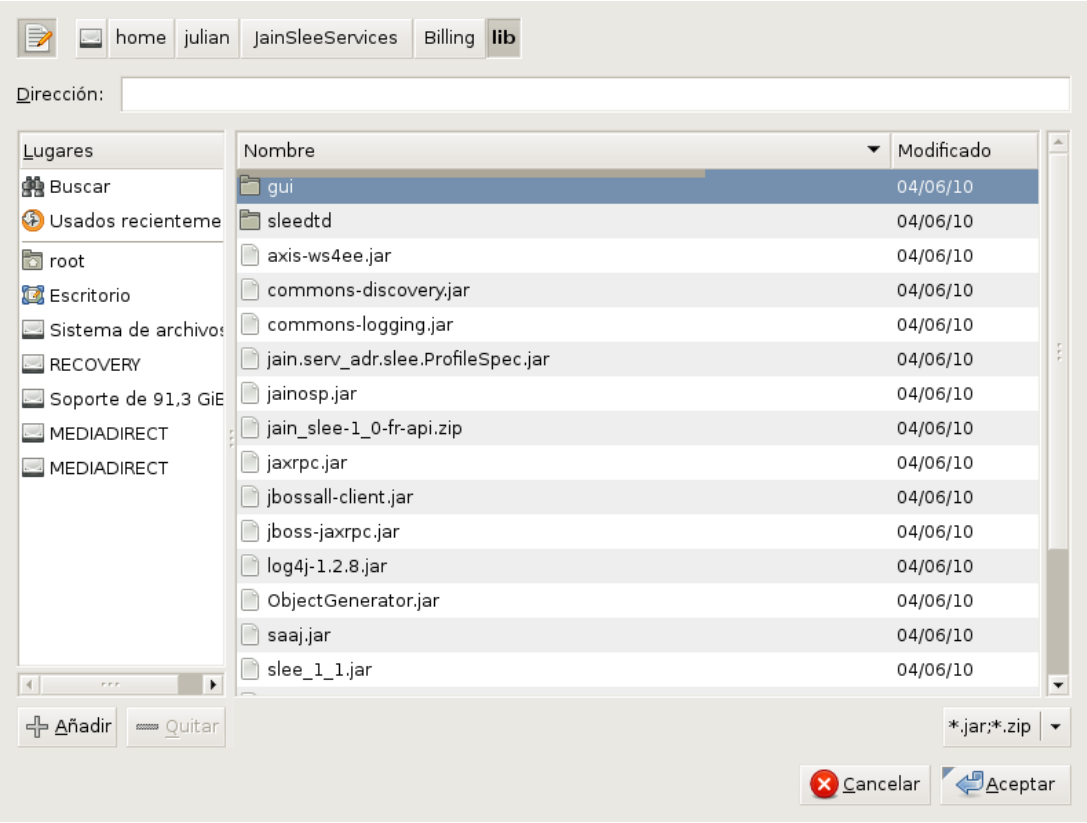

Figura 64. *Selección de archivos librería de Alcatel SCE (propia).*

Debemos observar en la pantalla la siguiente figura:

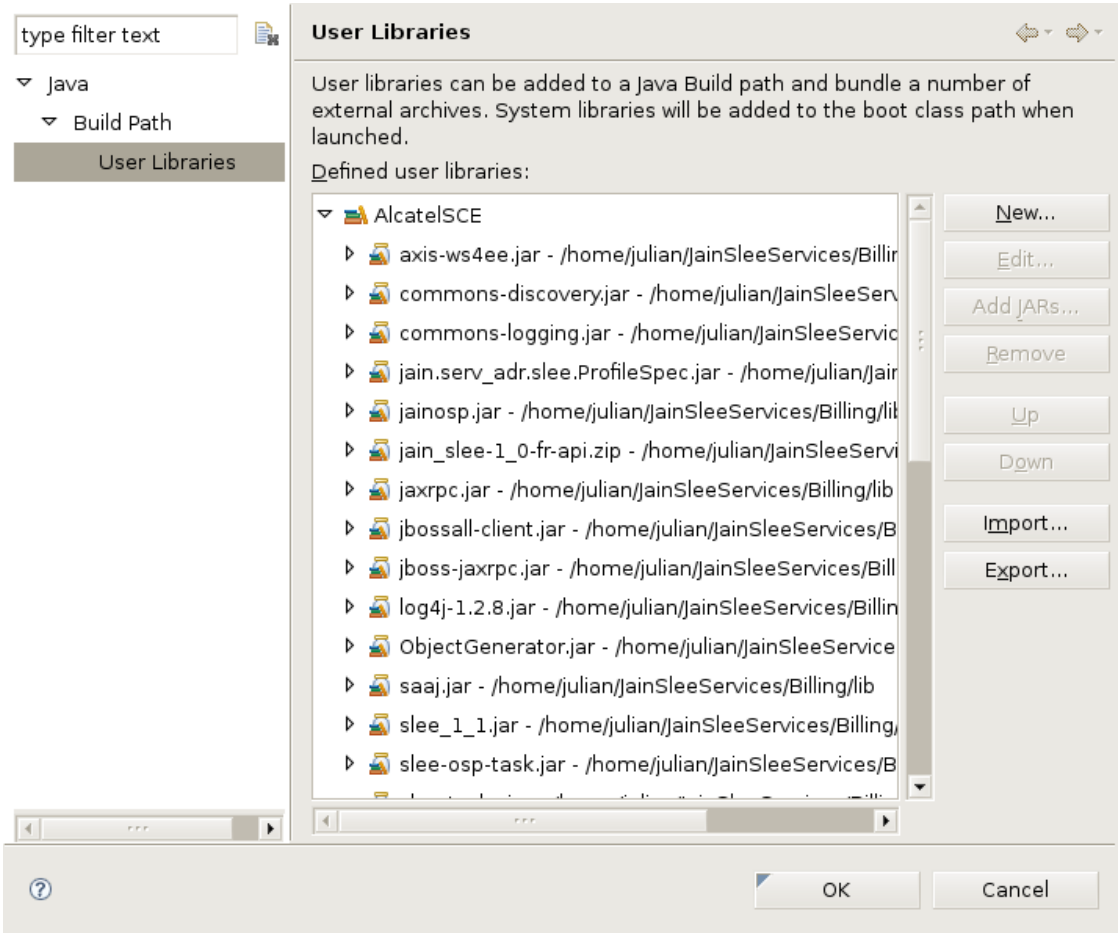

Figura 65. *Archivos .jar de la librería de Alcatel SCE (propia).*

Luego finalizamos la creación de la librería (figura 66).

- *Click OK → Check AlcatelSCE → Finish*

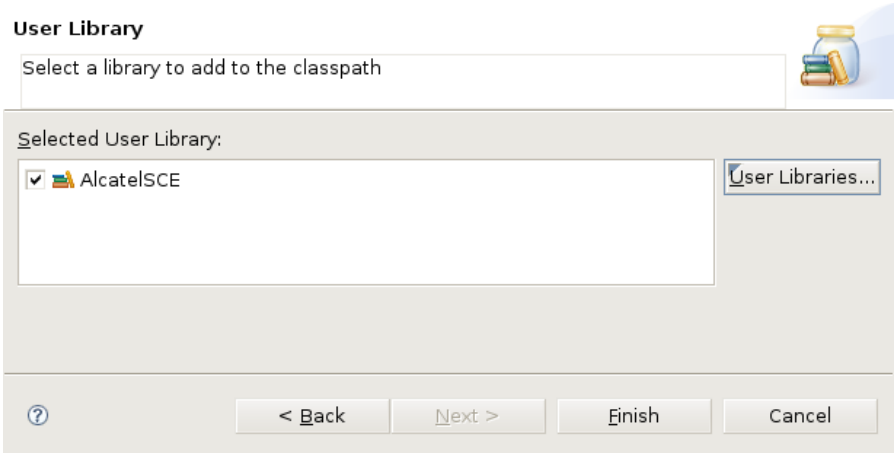

Figura 66. *Finalizar creación librería Alcatel SCE (propia).*

Finalmente en la figura 67, podemos observar nuestro nuevo proyecto, en el cual se puede crear componentes JAIN SLEE tanto de EclipSLEE como de Alcatel-Lucent SCE-SE.

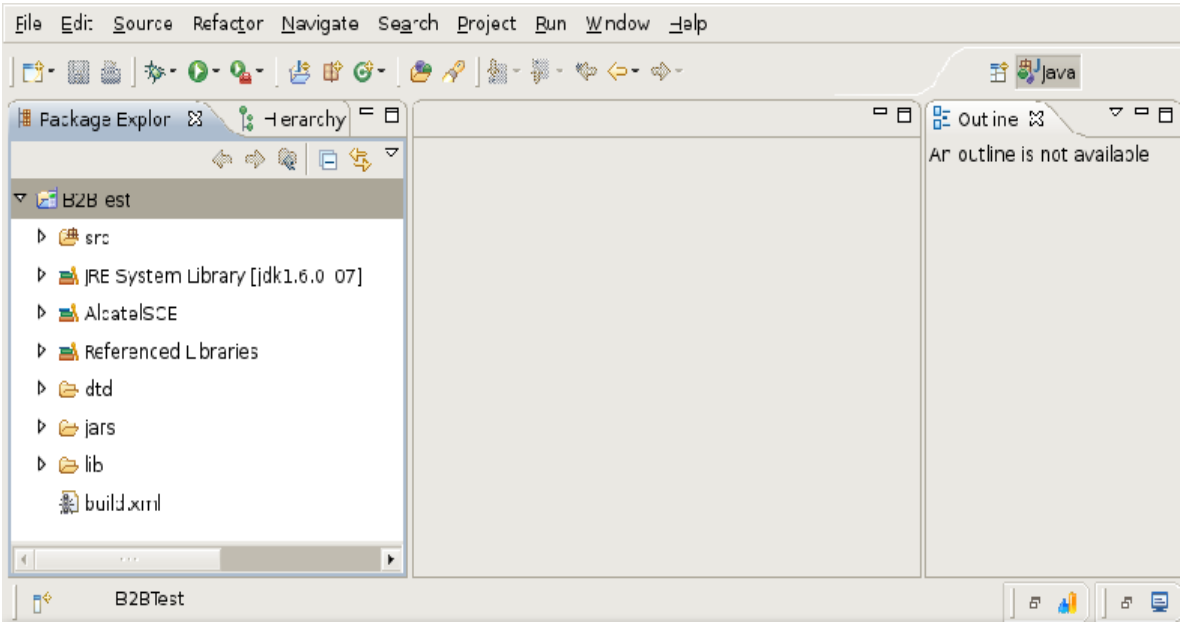

Figura 67. *Proyecto que integra EclipSLEE y Alcatel SCE (propia).*