Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

SERVICIO DE ACCESO ABIERTO PARA REDES DE PRÓXIMA GENERACIÓN BASADO EN PARLAY/OSA

Tesis para optar al Título de:
Ingeniero Civil en Informática

Profesor Patrocinante:
Sr. Luís Hernán Vidal Vidal.
Ingeniero Civil en Informática
Licenciado en Ciencias de la Ingeniería

Profesor Co-Patrocinante:
Sr. Jorge Morales Vilugrón
Ingeniero de Ejecución en Electricidad
Diplomado en Ingeniería, especialidad en Electricidad

Profesor Informante:
Sr. Luis Alberto Álvarez González
Ingeniero Civil Electricista
Magíster en Ingeniería Informática
D.E.A. Técnicas Informativas Avanzadas

SEBASTIÁN IGNACIO CASTILLO CASTRO

VALDIVIA – CHILE

2009

Universidad Austral de Chile
Instituto de Informática

Valdivia, 13 de Enero de 2009.

De : Luis Hernán Vidal Vidal.
A : Sr. Juan Pablo Salazar F.
Director de Escuela de Ingeniería Civil en Informática.
Ref. : Informa Calificación Trabajo de Titulación.

MOTIVO: Informar revisión y calificación del Proyecto de Título "Servicio de Acceso Abierto para redes de Próxima Generación basado en Parlay/OSA.", presentado por el alumno SEBASTIÁN IGNACIO CASTILLO CASTRO, que refleja lo siguiente:

Se logró el objetivo planteado que permitió desarrollar un servicio de acceso abierto basado en la especificación Parlay/OSA utilizando CORBA y/o WSDL.

El esfuerzo constante, junto con la dedicación al tema son aspectos destacables durante el proceso del trabajo realizado.

Cumplimiento del objetivo propuesto.	7,0
Satisfacción de alguna necesidad.	6,5
Aplicación del método científico.	7,0
Interpretación de los datos y obtención de conclusiones.	7,0
Originalidad.	7,0
Aplicación de criterios de análisis y diseño.	7,0
Perspectivas del trabajo.	6,7
Coherencia y rigurosidad lógica.	7,0
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración.	7,0
Evaluación Tesis.	**6,9**

Por todo lo anterior expuesto califico el trabajo de titulación del Sr. SEBASTIÁN IGNACIO CASTILLO CASTRO con nota 6,9 (seis coma nueve).

Sin otro particular, se despide atentamente.

Ing. Luis Hernán Vidal Vidal.
Profesor Instituto de Informática.
Facultad de Ciencias de la Ingeniería.
Universidad Austral de Chile.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA
FACULTAD DE CIENCIAS DE LA INGENIERIA

VALDIVIA, 2 de abril de 2009

De: JORGE ANTONIO MORALES VILUGRON
INFORMANTE

A: Juan Pablo Salazar Fernández
Director

Escuela de Ingeniería Civil en Informática

Ref.: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación "Servicio de Acceso Abierto para redes de Próxima Generación basado en Parlay/OSA", presentado por el/la alumno(a) sr./sra./srta. SEBASTIÁN IGNACIO CASTILLO CASTRO, mi evaluación del mismo es la siguiente:

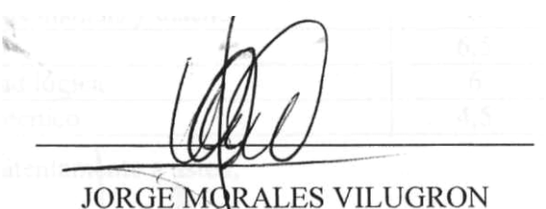
Nota: 6,8 (Seis como ocho).

Fundamento de la nota:

Una tesis que en sus aspectos técnicos esta muy bien definida, falto mas empeño en obtener "respuestas" a la encuesta a proveedores y esto incide en alguna conclusión y la satisfacción de alguna necesidad.

Aspecto	**Evaluación**
Cumplimiento de objetivos	7.0
Satisfacción de alguna necesidad	6.5
Aplicación del método científico	6.5
Interpretación de los datos y obtención de conclusiones	6.5
Originalidad	7.0
Aplicación de criterios de análisis y diseño	7.0
Perspectivas del trabajo	7.0
Coherencia y rigurosidad lógica	7.0
Precisión del lenguaje técnico	7.0

Sin otro particular, saluda atentamente a usted,



JORGE MORALES VILUGRON
ACADEMICO
INSTITUTO DE INFORMATICA

Universidad Austral de Chile
Instituto de Informática

Valdivia, 2 de marzo del 2009.

De: Luis Alberto Álvarez González.
Informante

A: Juan Pablo Salazar Fernández.
Director Escuela de Ingeniería Civil en Informática

Ref.: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"SERVICIO DE ACCESO ABIERTO PARA REDES DE PRÓXIMA GENERACIÓN BASADO EN Parlay/OSA",** presentado por el estudiante **SR. SEBASTIAN IGNACIO CASTILLO CASTRO,** mi evaluación es la siguiente:

Nota: 6,0 (SEIS COMA CERO).

Fundamento de la nota:

Su Tesis de Titulación cumple con los objetivos específicos, sin embargo, el último no guarda relación con los anteriores y perfectamente pudo ser parte del primero. Falta mayor rigurosidad en el uso de tipos de letras, referencias a figuras entre otros.

Aspecto	**Evaluación**
Cumplimiento de objetivos	5
Satisfacción de alguna necesidad	6,5
Aplicación del método científico	6
Interpretación de los datos y obtención de conclusiones	6,5
Originalidad	7
Aplicación de criterios de análisis y diseño	6
Perspectivas del trabajo	6,5
Coherencia y rigurosidad lógica	6
Precisión del lenguaje técnico	4,5

Sin otro particular, saluda atentamente a usted,

Luis A. Álvarez González
Instituto de Informática

Dirección : General Lagos 2086 – Campus Miraflores- Valdivia – Chile
Fono: 56 63 221427 · Fax: 56 63 293115 · email: instituto@inf.uach.cl

Agradezco y dedico esta tesis a toda mi familia, en especial a mis padres, quienes con gran esfuerzo y sacrificio lograron darme una educación de calidad, entregándome apoyo incondicional en todo momento. A mis queridas hermanas que siempre estuvieron conmigo en este proceso esperando lo mejor de mí.

Considero el presente trabajo de tesis sólo como un último paso, y de carácter simbólico, para cumplir esta meta tan esperada. La verdadera formación como profesional sin duda creo haberla logrado en el transcurso de todos mis años de estudio, y es por ello que sinceramente deseo manifestar mis agradecimientos a todo el cuerpo docente, paradocente y personal administrativo, quienes de una u otra forma aportaron a mi formación a lo largo de todo este importante proceso.

Quisiera agradecer de forma especial al profesor Luís Vidal Vidal por haber depositado confianza absoluta en mi persona y en mis conocimientos, lo cual me permitió llegar a tomar decisiones relevantes en este trabajo las que siempre fueron bien apoyadas por él, logrando con ello desarrollar lo que realmente he pretendido plasmar en esta tesis.

Finalmente no quisiera dejar de mencionar a todos mis amigos y compañeros con quienes en el transcurso de la carrera avanzamos aportando cada cual lo suyo para lograr nuestros objetivos, deseándoles el mejor de los éxitos en sus realizaciones.

ÍNDICE DE CONTENIDO

RESUMEN ..... **VIII**

ABSTRACT ..... **IX**

CAPÍTULO I: INTRODUCCIÓN **1**

1.1 GENERALIDADES 1

1.2 ANTECEDENTES 2

1.3 IMPORTANCIA Y NATURALEZA DEL ESTUDIO 3

1.4 ESTRUCTURA DE LA TESIS 3

1.5 OBJETIVOS GENERALES Y ESPECÍFICOS 6

1.5.1 Objetivo General *6*

1.5.2 Objetivos Específicos *6*

CAPÍTULO II: CAMINO HACIA LA CONVERGENCIA DE REDES DE TELECOMUNICACIONES **7**

2.1 REDES DE PRÓXIMA GENERACIÓN 7

2.1.1 Transición hacia un enfoque horizontal *7*

2.1.2 Definición de Redes de Próxima Generación (NGN). *8*

2.1.3 Tendencias *10*

2.1.4 Tecnologías Habilitantes para la convergencia *13*

2.2 PARLAY/OSA Y PARLAY-X: UN NUEVO ENFOQUE EN LOS NEGOCIOS 17

2.2.1 Nuevos modelos de negocio: abriendo la red a terceros *17*

2.2.2 Las ventajas de Parlay/OSA y Parlay-X *19*

CAPÍTULO III: EL ESTÁNDAR **23**

3.1 ESTÁNDAR PARLAY/OSA 23

3.1.1 Arquitectura conceptual *23*

3.1.1.1 La aplicación cliente 24

3.1.1.2 Los SCSs o Servidores de Capacidades de Servicio. 25

3.1.1.3 El framework. 26

3.1.1.4 Interacción entre los tres componentes. 27

3.1.2 Servicios disponibles en el gateway Parlay/OSA 5.0 ..... *29*

3.1.3 Especificaciones de las APIs de Parlay/OSA *30*

3.1.4 Esquemas de versionamiento *30*

3.1.5 APIs de Parlay/OSA: diseño bajo UML *31*

3.1.5.1 Tecnología CORBA IDL 32

3.1.5.2 Tecnología Java™ 32

3.1.5.3 Tecnología WSDL 32

3.1.6 Documentos de especificación para Parlay/OSA 5.0 ..... *33*

3.2 PARLAY-X 34

3.2.1 Arquitectura conceptual *34*

3.2.2 Características de Parlay-X *35*

3.2.3 Capacidades de servicios en Parlay-X *36*

CAPÍTULO IV: ERICSSON SDK PARA SERVICIOS WEB DE TELECOMUNICACIONES. 38

4.1 DESCRIPCIÓN DEL KIT DE DESARROLLO 38

4.1.1 El emulador *38*

4.1.2 Componentes Java SE *40*

4.2 SOFTWARE REQUERIDO POR EL EMULADOR 41

4.3 INTERFAZ DE USUARIO DEL EMULADOR 42

4.3.1 Interfaz gráfica principal *42*

4.3.2 Sección de configuración *44*

4.3.3 Terminales móviles de prueba *45*

4.3.4 Servicio de localización de terminales *49*

CAPÍTULO V: PAM, UN SERVICIO PARLAY/OSA. 52

5.1 PAM: PRESENCE AND AVAILABILITY MANAGEMENT. 52

5.1.1 Introducción *52*

5.1.2 Conceptos claves *53*

5.1.3 Interfaces de clases y diagramas UML *54*

5.1.3.1 Interfaces de clases que componen a PAM 54

5.1.3.2 Esquema de colores y nombres para las clases de interfaces 55

5.1.3.3 SCF PAM Provisioning: Interfaces de clases y diagrama UML 55

5.1.3.4 SCF PAM Access: Interfaces de clases y diagrama UML 57

5.1.3.5 SCF PAM Event: Interfaces de clases y diagrama UML 59

5.2 UN FRAMEWORK PARA PARLAY/OSA 60

5.2.1 Descripción *60*

5.2.2 Tecnologías empleadas *61*

5.2.3 Configuraciones para la puesta en marcha del framework *61*

5.2.4 Archivo buid.xml y preparación de la base de datos *63*

5.2.5 Puesta en marcha del framework y del servidor de nombres *64*

5.2.5.1 Clase IpInitial: interfaz principal del framework 65

5.3 IMPLEMENTACIÓN DEL SERVICIO PAM 66

5.3.1 Herramientas y tecnologías utilizadas *66*

5.3.2 Preparación del entorno de desarrollo y tareas preliminares *68*

5.3.2.1 Creación del proyecto y estructura de directorios 68

5.3.2.2 Archivos IDL requeridos 69

5.3.2.3 Creación del archivo build.xml y compilación IDL-to-Java 69

5.3.2.4 Archivos de configuración 71

5.3.3 Interfaces y clases Java generadas a partir de los archivos IDL *72*

5.3.4 Esquema jerárquico de paquetes: organización de clases e interfaces implementadas *74*

5.3.5 CorbaServiceImpl.java: implementación de métodos CORBA usando JacORB. *75*

5.3.6 Interfaces PAM implementadas *76*

5.3.6.1 Procesamiento de información de presencia y disponibilidad 78

5.3.7 Persistencia de datos: mapeo Objeto-Relacional usando la herramienta Hibernate *79*

5.3.8 Interacción PAM-Framework: Interfaces implementadas ... 82
5.3.9 Preparación del entorno de ejecución del servicio PAM ... 86
5.3.9.1 Deploy de PAM usando la herramienta Ant ... 86
5.3.9.2 Registro de los SCFs en la base de datos ... 87
5.3.9.3 Creación del entorno de ejecución ... 88
5.3.10 Puesta en marcha del servicio PAM ... 88
5.4 IMPLEMENTACIÓN DE APLICACIONES DE PRUEBA ... 91
5.4.1 Prueba del la SCF PAM Event. ... 93
5.4.2 Prueba del las SCFs PAM Provisioning y PAM Access ... 95

CAPÍTULO VI: APLICACIÓN PARLAY-X: LOCALIZACIÓN DE USUARIOS A TRAVÉS DE GOOGLE MAPS ... 98

6.1 DESCRIPCIÓN DE LA APLICACIÓN ... 98
6.2 "TERMINAL LOCATION": SERVICIO WEB DE PARLAY-X ... 98
6.2.1 Interfaz gráfica del servicio de localización del emulador ... 98
6.3 API DE GOOGLE MAPS ... 100
6.4 IMPLEMENTACIÓN DE LA APLICACIÓN PARLAY-X ... 100
6.5 EJECUCIÓN DE LA APLICACIÓN PARLAY-X ... 103

CAPÍTULO VII: SITUACION ACTUAL DE INSERCIÓN DEL ESTANDAR A NIVEL NACIONAL ... 106

7.1 ENCUESTA REALIZADA Y RESULTADOS ... 106
7.2 ANÁLISIS ... 107

CONCLUSIONES ... 108

REFERENCIAS ... 110

ÍNDICE DE TABLAS

Tabla 1. Desalineamiento en las versiones de Parlay para una misma especificación. 31

Tabla 2. Lista de documentos que especifican las APIs de Parlay/OSA versión 5.0 33

Tabla 3. Servicios Web de Parlay-X implementados en el emulador..... 43

Tabla 4. Jerarquía de paquetes de clases generadas desde los IDLs, y su contenido..... 73

Tabla 5. Jerarquía de paquetes de clases de interfaces implementadas en PAM..... 75

ÍNDICE DE FIGURAS

Figura 1. Estructura de la tesis. 5

Figura 2. Redes de servicios simples vs redes de múltiple servicios. 7

Figura 3. El papel del protocolo IP en las Redes de Próxima Generación. 8

Figura 4. Crecimiento en redes fijas, móviles, e Internet, en cuanto a la cantidad de suscriptores (en billones entre los años 1996 y 2006). 11

Figura 5. Convergencia: transformación de la industria. 12

Figura 6. Tendencias en la convergencia de redes de telecomunicaciones. 12

Figura 7. Evolución de las telecomunicaciones. 13

Figura 8. SIP, IMS y Parlay/OSA integrados bajo una misma arquitectura separada en capas. 15

Figura 9. SIP, Componentes IMS y Parlay/OSA integrados bajo una misma arquitectura. 17

Figura 10. Parlay/OSA y los nuevos modelos de negocio. Un nuevo jugador entra el mercado de los servicios. 18

Figura 11. Cantidad de desarrolladores según la interfaz utilizada para acceder a las funcionalidades de una red de telecomunicaciones. En el gráfico, 100s, 1000s, 1000000s indica, cientos, miles, millones respectivamente. 21

Figura 12. Arquitectura de Parlay/OSA 24

Figura 13. Aplicación cliente interactuando con la red, mediante una petición del usuario final. 25

Figura 14. Servidor de Capacidad de Servicio interactuando con la red, a petición de la aplicación cliente. 26

Figura 15. Registro y descubrimiento de servicios (SCFs), y uso de éstos por parte de aplicaciones clientes. 27

Figura 16. Arquitectura de Parlay-X y su relación con Parlay/OSA. 35

Figura 17. Emulador de Servicios Web para redes de telecomunicaciones interactuando con una aplicación. 39

Figura 18. Capa del SDK de Ericsson para establecer comunicación entre una aplicación y los Servicios Web de Parlay-X 40

Figura 19. Interfaz gráfica principal del emulador. 42

Figura 20. Modo de compatibilidad de Parlay-X. 44

Figura 21. Emulación de una excepción de Parlay-X 45

Figura 22. Interfaz de emulación de terminales. 46

Figura 23. Lista de terminales emulados. 46

Figura 24. Interfaces gráficas de los terminales emulados. 47

Figura 25. Estados posibles de los terminales móviles. 47

Figura 26. Opciones de "Messaging" y "Multimedia", y agregar contenido multimedia al terminal. 48

Figura 27. Envío de un mensaje SMS desde un terminal a otro. 49

Figura 28. Configuración de los parámetros del mapa utilizado en la emulación de la localización de terminales. 50

Figura 29. Mapa utilizado en la emulación de localización de terminales móviles 51

Figura 30. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Provisioning. 57

Figura 31. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Access. 59

Figura 32. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Event. 60

Figura 33. Ejecución del target "prepare-db" del archivo "*build.xml*", utilizando Ant. 63

Figura 34. El servidor de nombres de CORBA en ejecución. 64

Figura 35. Puesta en marcha del framework de Parlay/OSA. 65

Figura 36. Referenciación de la interfaz del *framework* "`IpInitial`" en el servidor de nombres. 66

Figura 37. Compilación IDL-to-Java realizada con la herramienta Ant. 71

Figura 38. Jerarquía en el esquema de paquetes de clases generadas. 72

Figura 39. Esquema de paquetes de clases e interfaces implementadas en el proyecto PAM. 74

Figura 40. Esquema de paquetes de las clases implementadas creado para el proyecto PAM. 78

Figura 41. Lista de clases creadas para mantener la persistencia de datos. 80

Figura 42. Modelo relacional utilizado en el mapeo Objeto-Relacional. 82

Figura 43. Paquetes que contienen la clase *ServiceMain.java* 83

Figura 44. Autenticación y registro de servicios (SCFs) en el *framework*. Fuente [Moe02]. 84

Figura 45. Deploy de PAM usando la herramienta ANT. Se muestran sólo los primeros pasos realizados después de ejecutar "ant deploy" 87

Figura 46. Registro de los SCFs en la base de datos usando la herramienta Ant 88

Figura 47. Puesta en marcha de PAM SCF Access mediante la ejecución del script *serviceAccess.bat* 90

Figura 48. Puesta en marcha de PAM SCF Provisioning mediante la ejecución del script *serviceProvisioning.bat* 90

Figura 49. Puesta en marcha de PAM SCF Event mediante la ejecución del script *serviceEvent.bat* 91

Figura 50. La aplicación “consumer” en ejecución, quedando en espera a la notificación de eventos desde la pasarela de Parlay/OSA. 93

Figura 51. Aplicación cliente “*Consumer*”: estructura de árbol que reacciona a los eventos lanzados al crear agentes, identidades y atributos. 94

Figura 52. Creación de identidades, agentes y algunos atributos de ejemplos ejecutada por la aplicación cliente “*Application*”. 94

Figura 53. Aplicación cliente “Gui”: ingreso de atributos de presencia para una identidad 96

Figura 54. Aplicación cliente “Gui”: ingreso de atributos de presencia para un agente 96

Figura 55. Aplicación cliente “Gui”: opciones disponibles en el menú “Provisioning”. 97

Figura 56. Aplicación cliente “Gui”: opciones disponibles en el menú “Access”. 97

Figura 57. Interfaz gráfica del servicio de localización del emulador mostrando un mapa de una zona de la ciudad de Valdivia. 99

Figura 58. Entorno de desarrollo Eclipse usado para implementar la aplicación Parlay-X. 101

Figura 59. Componentes y tecnologías empleadas en la implementación de la aplicación, junto con la interacción entre los componentes. 102

Figura 60. Carga de la aplicación Web creada en el Servidor de Aplicaciones Java Glassfish. 104

Figura 61. Fig. izquierda, el emulador y su interfaz gráfica de localización de terminales; fig. derecha, aplicación Parlay-X para la localización de usuarios. 105

Figura 62. Fig. izquierda, el emulador y la nueva ubicación del terminal; fig. derecha, aplicación Parlay-X desplegando la nueva ubicación. 105

RESUMEN

Las telecomunicaciones y las tecnologías de la información, dos mundos que surgieron de forma independiente, tienden cada vez más a la convergencia formando una nueva y gran arquitectura de red. Surge así el término de Redes de Próxima Generación, cuyo principal enfoque es la unificación de todas las redes de telecomunicaciones presentes en cualquiera de sus cuatro formas de acceso: inalámbricas, móviles, por fibra y por cable. Debido a la creciente necesidad de conectividad y movilidad para los usuarios finales, los servicios deberán estar disponibles en áreas de cobertura lo más amplias posible desde cualquier medio de acceso. Parlay/OSA es el elemento habilitador que hace posible que el escenario anterior sea llevado a la realidad. Las especificaciones de Parlay/OSA definen una arquitectura que posibilita a los desarrolladores de aplicaciones de servicio hacer uso de las funcionalidades de las redes de telecomunicaciones a través de APIs estandarizadas y abiertas, dando lugar de esta manera a nuevos modelos de negocios en donde las aplicaciones pueden ser desarrolladas y provistas por empresas fuera del dominio de los operadores de redes.

El presente proyecto tiene como objetivo principal presentar la implementación de "*Presence and Availability Management*", un servicio de acceso abierto basado en la especificación Parlay/OSA, utilizando tecnologías Java y CORBA. Previamente se realizará una descripción teórica de Parlay/OSA, y del estándar Parlay-X el cual define interfaces para acceder a las funcionalidades de la red a través de Servicios Web, describiendo también el rol que juegan estos estándares en las Redes de Próxima Generación, y los beneficios que trae consigo su incorporación en las compañías de telecomunicaciones. Se mostrará además, la implementación de una aplicación Parlay-X que ofrece un servicio de localización de un usuario móvil determinado a través de una página Web utilizando la API de Google Maps.

Por último se efectuará un análisis de la situación actual de inserción del estándar Parlay/OSA y Parlay-X a nivel nacional, a través de la realización de una encuesta a las principales compañías nacionales de telecomunicaciones.

ABSTRACT

Telecommunications and information technology, two worlds that appeared in an independent way, are tending increasingly to converge in a new major network architecture. This way the term Next Generation Network arises, whose main focus is the unification of all the networks in any of the four types of access: wireless, mobile, fiber and cable. Due the growing need for connectivity and mobility for final users, the services should be available in coverage areas as wide as possible from any way of access. Parlay/OSA is the enabler that allows the previous scenario brought to reality. The specifications of Parlay/OSA defines an architecture that enables application developers to use the service functions of telecommunications networks via standard and open APIs, resulting in this way new business models where the applications can be developed and provided by companies outside the domain of network operators.

The main goal of this project is to present the implementation of "Presence and Availability Management", an Open Service Access based on the Parlay/OSA specification, using Java and CORBA technologies. Previously it will be carried out a theoretical description of Parlay/OSA and Parlay-X standard which defines interfaces to access the network functionality through Web Services, also describing the role played by these standards in the Next Generation Network, and the benefits that they bring to telecommunications companies. It will appear in addition the implementation of a Parlay-X application that provides a service for locating a mobile user over a given Web page using the Google Maps API.

Finally, there will be carried out an analysis of the current situation of integration of the Parlay/OSA and Parlay-X standard at national level, across the accomplishment of a survey to the major national telecommunications companies.

CAPÍTULO I: INTRODUCCIÓN

1.1 Generalidades

Las telecomunicaciones están experimentando uno de los procesos de cambio más fuertes y decisivos conocidos hasta ahora. Después de la falta de regulación, de la modernización y de las privatizaciones que comenzaron a mediado de los ochenta en todo el mundo, las telecomunicaciones alcanzaron un punto de crisis en el año 2000 [Fra07], debido, principalmente, a los modelos regulatorios que dieron más importancia al mercado, las tarifas y la apertura y no a la calidad de los servicios y universalidad de éstos.

Servicios de voz, localización, MMS, SMS, juegos, entre otros, han ido transformando las redes de telecomunicaciones en verdaderas marañas de alambres, con infinitas integraciones. Por otro lado, la creciente demanda de los usuarios por nuevos servicios y aplicaciones crea una gran oportunidad para extender el negocio. Organizar toda esta estructura es un gran desafió y para intentar dar solución a esta problemática a un nivel de implementación de servicios, los operadores, desarrolladores, empresas de telecomunicaciones y de las tecnologías de la información se unieron para definir las APIs de Parlay/OSA a través de la creación del Forum de Tecnología Parlay, un consorcio abierto que actualmente cuenta con más de ochenta miembros, entre los cuales, los más importantes son IBM, ERICSSON, Sun Microsystems, Siemens, Nokia, AT&T, Fujitsu, HP, Cisco Systems y Oracle. [Par05a].

Las APIs de Parlay/OSA son especificadas por el grupo Parlay en conjunto con ETSI[1] y 3GPP[2], y están disponibles de forma abierta en la red para quien se interese en ellas.

En este proyecto se presenta la implementación del servicio PAM (Manejo de Presencia y Disponibilidad) cuya especificación se encuentra definida en el documento "parte 14: *Presence and Availability Management*" [Ets05b] de las APIs de Parlay/OSA versión 5.0. La implementación de PAM tuvo lugar en un laboratorio de Sistemas Distribuidos realizado por el estudiante tesista, durante una beca de intercambio en el periodo Ago05/Jul06, en la universidad alemana RWTH-Aachen[3], junto con dos estudiantes de la carrera Ingeniería en Telecomunicaciones (Universidad Politécnica de

[1] European Telecom Standard Institute
[2] 3rd.Generation Partnership Project
[3] http://www.rwth-aachen.de

Madrid). La presente tesis, describe todo el trabajo realizado por el grupo, y se enfoca principalmente en la implementación del servicio PAM, tarea desempeñada en gran parte por el alumno tesista. Los dos otros integrantes del grupo, cumplieron las labores de implementación de aplicaciones de prueba del servicio y de interconexión con el *framework* de Parlay/OSA en donde el estudiante tesista participó en un menor grado.

El estándar Parlay-X, que especifica sus APIs a través de Servicios Web, también fue considerado como parte importante del presente proyecto de tesis, debido a la gran popularidad que ha tenido en los últimos años las tecnologías SOAP y XML, naciendo como una iniciativa para simplificar el modelo de Parlay/OSA, con la finalidad de que pueda ser usado con mayor facilidad y dinamismo por profesionales del mundo de la programación que no tengan que ser necesariamente técnicos en redes de telecomunicaciones.

1.2 Antecedentes

Las especificaciones de Parlay/OSA no describen cómo éstas deben ser implementadas, lo que permite a los desarrolladores flexibilidad en la implementación de las APIs. Actualmente algunas empresas, principalmente europeas, se dedican a implementar el *Gateway* de Parlay/OSA compuesto por un *framework* y por los servicios especificados por las APIs, entre ellos, PAM. Otras empresas, también europeas en su mayoría, se dedican al desarrollo de aplicaciones y servicios basados en el estándar Parlay/OSA y Parlay-X (Servicios Web) [Para]. En Latinoamérica sólo se registran tres empresas de telecomunicaciones que han adoptado el estándar Parlay/OSA como plataforma base para la implementación de sus servicios y aplicaciones. Estas empresas son Brasil-Telecom en Brasil, Telmex de México y Comcel de Colombia [Fre]. En cuanto a proyectos y tesis realizadas sobre Parlay/OSA, éstas se han llevado a cabo principalmente en algunas universidades europeas, y en América se registra un proyecto realizado por el Instituto Internacional para las Telecomunicaciones, en Montereal, Canadá [Parb]. Hasta ahora, a nivel nacional no se encuentran registros[4] de proyectos, tesis, *papers* o trabajos realizados sobre el estándar Parlay/OSA y Parlay-X, ni a nivel académico, ni a nivel comercial.

---

[4] Basado en búsquedas a través de Internet.

1.3 Importancia y Naturaleza del Estudio

Las APIs de Parlay/OSA y Parlay-X entregan a los desarrolladores la posibilidad de crear una gran cantidad de innovadoras aplicaciones telemáticas mediante la combinación de los servicios entregados por las redes de telecomunicaciones que son soportados por las especificaciones de las APIs. La posibilidad está limitada sólo por la imaginación de los desarrolladores.

La principal motivación de este proyecto surge debido a que a nivel nacional no se han realizado trabajos relacionados principalmente con el estándar Parlay/OSA. Además, Chile es un gran consumidor de tecnología, sobre todo en el ámbito de las telecomunicaciones, lo indican factores como, por ejemplo, el 84% de los chilenos que poseen un celular, según informe dado a conocer por la Subtel en Diciembre de 2007 [Sub07]. Parlay/OSA trae un nuevo modelo de negocio para las empresas de telecomunicaciones en donde todos ganan: operadores, usuarios, y desarrolladores de servicios. Aparecen nuevas fuentes de ingresos para los operadores debido a que finalmente se llega a un incremento en el tráfico causado por el enriquecimiento de la oferta de servicios. Además se obtienen otros beneficios como por ejemplo la reducción de la rotación de clientes; reducción del tiempo para poner en el mercado nuevos servicios; generación de mayores ingresos en servicios tradicionales por medio de la integración de éstos con nuevos servicios y la inclusión de mejorías, como por ejemplo, un servicio de correo de voz que envía SMS; expansión de la cadena de valor del operador, penetrando en nuevos segmentos y mercados comerciales; y haciendo posible nuevas estrategias de mercado por medio del desarrollo rápido y económico de nuevos servicios, a través del aprovechamiento de aplicativos de terceros y del uso de proveedores de servicios de aplicativos externos fuera del dominio del operador de telecomunicaciones [Par05a]. Todos estos beneficios podrían concretarse con la incorporación de este estándar en las compañías de telecomunicaciones nacionales, siendo esto el principal factor de motivación del presente trabajo.

1.4 Estructura de la Tesis

La presente Tesis se compone de seis capítulos, los cuales pueden ser clasificados en dos partes:

La primera parte tiene relación con el estado del arte del tema a tratar. Se comienza con una introducción y aspectos generales (**Capítulo I**), señalando el lugar en donde se realizó la implementación del servicio Parlay/OSA e indicando algunos antecedentes sobre los trabajos, tesis realizadas, *paper* y desarrollos existentes de productos basados en el estándar Parlay/OSA, junto con entregar una visión general sobre la motivación e importancia de esta tesis. Además se indican los objetivos generales y específicos del presente proyecto. Más adelante, en el **Capítulo II** se introducen los conceptos requeridos para comprender la evolución de las redes de telecomunicaciones hacia la unificación de éstas, surgiendo el término de Redes de Próxima Generación, junto con las tecnologías involucradas en la convergencia de redes, entre ellas, Parlay/OSA y Parlay-X, estándares descritos en detalle en el **Capítulo III**, en donde se explica el funcionamiento de la pasarela (*gateway*) Parlay/OSA y de Parlay-X consistentes en un capa intermedia de software utilizada como interfaz para acceder de forma estandarizada a las funcionalidades de las redes de telecomunicaciones. Se describen todos los elementos requeridos para que una aplicación pueda utilizar una funcionalidad de red determinada, junto con las tecnologías involucradas en este proceso.

La segunda parte de la tesis consiste en la aplicación práctica de lo abarcado principalmente en el capitulo III. Esta segunda parte comprende de un **Capítulo V** el cual tiene por objetivo principal presentar en detalle la implementación realizada de un servicio de “Manejo de Disponibilidad y Presencia” bajo las especificaciones de las APIs de Parlay/OSA versión 5.0 diseñadas fundamentalmente en base a diagramas UML, realizando previamente una descripción del servicio y de los conceptos claves involucrados. Además, con la finalidad de probar el funcionamiento del servicio Parlay/OSA implementado, en este mismo capítulo se presentan tres aplicaciones de prueba implementadas para este propósito. Con el objetivo de realizar una demostración práctica del potencial de las APIs de Parlay-X, en el **Capítulo VI** se describe la implementación de una aplicación Web consistente en un servicio de localización de usuarios, la cual obtiene información de localización de forma emulada a través de un Emulador de Parlay-X desarrollado por Ericsson (**Capítulo IV**) en el cual se encuentran implementados algunos Servicios Web especificados en Parlay-X versión 2.1.

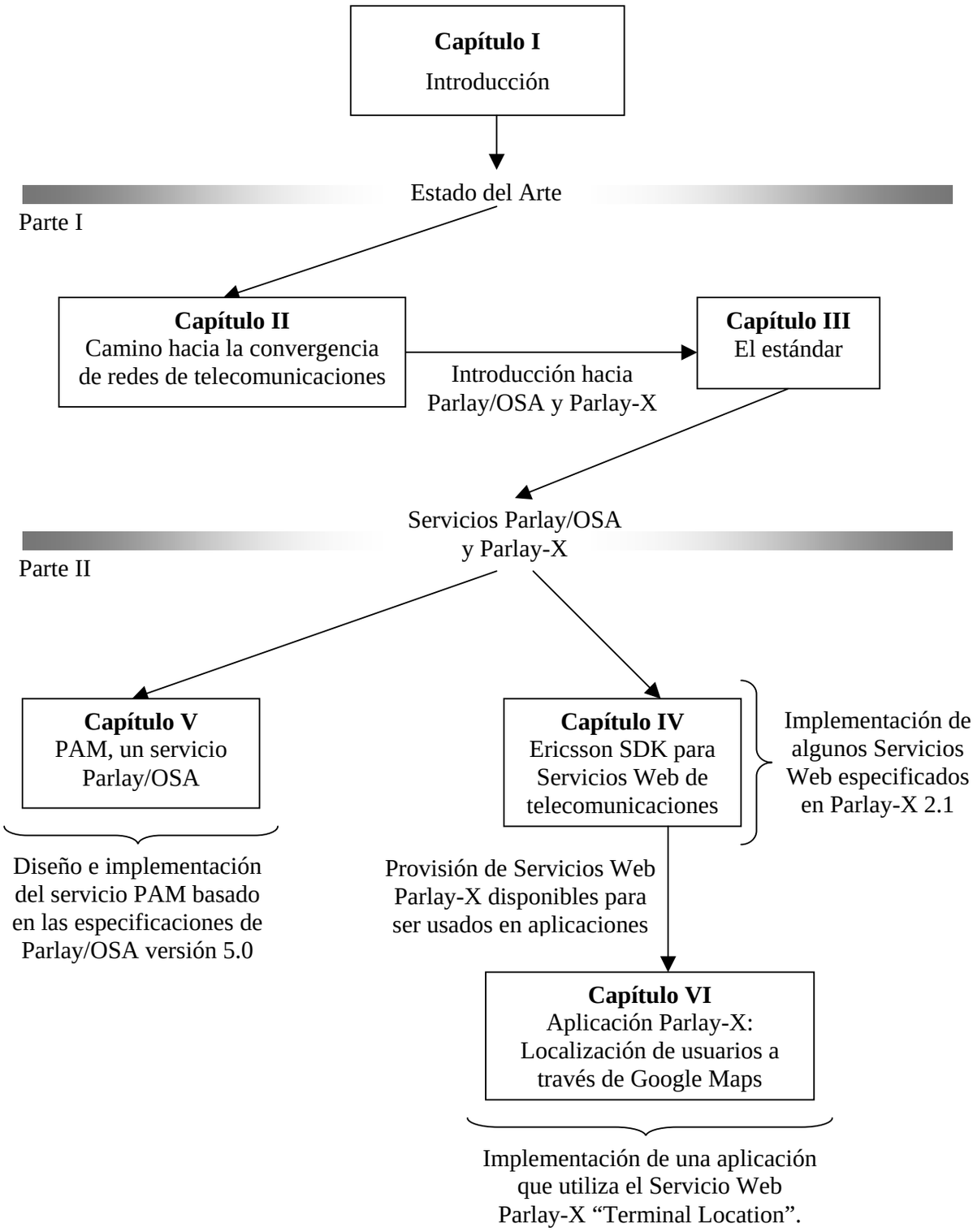


Figura 1. Estructura de la tesis.

1.5 Objetivos Generales y Específicos

1.5.1 Objetivo General

- Desarrollar un servicio de acceso abierto basado en la especificación Parlay/OSA utilizando CORBA[5] y/o WSDL[6].

1.5.2 Objetivos Específicos

- Describir el estándar Parlay/OSA y Parlay-X identificando su función en las empresas de telecomunicaciones.
- Definir servicio y describir diseño de operación basado en la API de Parlay/OSA.
- Implementar el servicio haciendo uso de un *framework* desarrollado bajo la especificación Parlay/OSA.
- Analizar situación actual de inserción del estándar Parlay/OSA y Parlay-X a nivel nacional.

[5] *Common Object Request Broker Architecture*
[6] *Web Service Definition Languaje*

CAPÍTULO II: CAMINO HACIA LA CONVERGENCIA DE REDES DE TELECOMUNICACIONES

2.1 Redes de Próxima Generación

2.1.1 Transición hacia un enfoque horizontal

En un comienzo, las redes fueron integradas verticalmente, y evolucionaron en dominios diferentes para dar soporte a servicios únicos tales como voz, datos y video [Gar08]. Para crear nuevos servicios, estos debían ser implementados directamente sobre las redes subyacentes, siendo necesario tener un conocimiento técnico profundo sobre los protocolos y señalizaciones de la red en la que el servicio iba a ser integrado, lo que finalmente derivaba en una lenta salida al mercado de éstos.

Con el pasar del tiempo, los usuarios finales comenzaron a requerir de nuevas funcionalidades que satisfagan sus necesidades de comunicación. La explosión de la popularidad de Internet demuestra de cierta forma el uso creciente, por parte de los usuarios, de nuevas formas de comunicarse con contenidos multimedia, y accesos a información instantánea, tanto en sus vidas personales como profesionales. La industria de las telecomunicaciones comienza a visualizar entonces la necesidad de la introducción de nuevos servicios y de un nuevo mecanismo para proveer estos servicios en las redes [Rau02]. Comienza así una transición de un enfoque vertical de despliegue de redes hacia uno horizontal, en donde la ejecución de servicios, el control y la conectividad pueden ser integrados horizontalmente a través de múltiples redes de acceso.

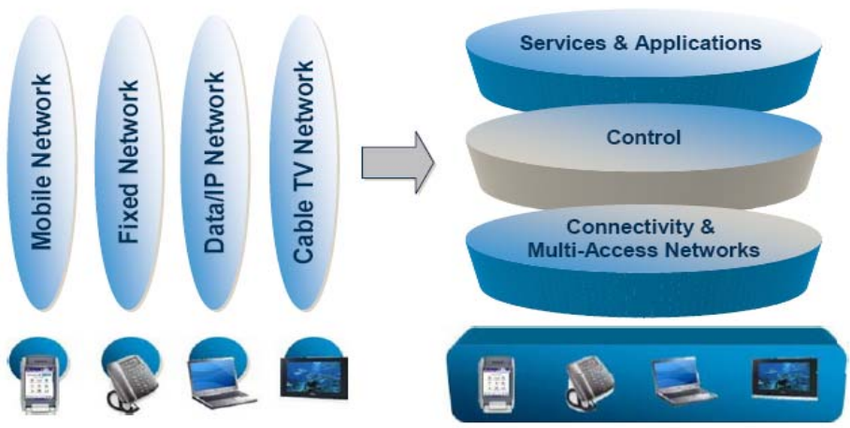


Figura 2. Redes de servicios simples vs redes de múltiple servicios.[7]

[7] Fuente: [Gar08]

Tecnologías como los *Softswitches* y el protocolo IP son los elementos que posibilitan la implementación una arquitectura de redes orientada al enfoque horizontal de capas [Eri06], llegando así a una primera aproximación de convergencia entre redes, en donde servicios como la voz y la televisión pasan a ser una aplicación más. El protocolo IP provee de acceso, a los usuarios finales, de contenidos en Internet con mayor riqueza y disponibles en múltiples formatos, los cuales, en un enfoque horizontal de redes, pueden ser accedidos a través de cualquiera de las redes subyacentes, ya sea a través de fijas de cobre y fibra, y redes móviles. Aparecen de esta manera nuevas capacidades de *streaming*, mensajería, control de llamadas, noticias, videollamadas, juegos, música, entretenimiento, servicios de presencia y disponibilidad, etc., presentes en una red global.

Figura 3. El papel del protocolo IP en las Redes de Próxima Generación.[8]

Esta nueva modalidad de acceder a las funcionalidades de la red de forma combinada, a través de distintas capas, e independiente del medio de acceso, dan origen a las Redes de Próxima Generación o NGN por sus siglas en inglés.

2.1.2 Definición de Redes de Próxima Generación (NGN).

La normalización y definición de las Redes de Próxima Generación han sido un esfuerzo conjunto de organizaciones normalizadoras internacionales tales como ITU, ETSI TISPAN, ATIS, OMA, IETF y 3GPP [Cit06].

[8] Fuente: [Gar06]

Una definición formal de Redes de Próxima Generación propuesta por la ITU [Itu04] es la siguiente:

"Red basada en paquetes que permite prestar servicios de telecomunicación y en la que se pueden utilizar múltiples tecnologías de transporte de banda ancha propiciadas por la QoS, y en la que las funciones relacionadas con los servicios son independientes de las tecnologías subyacentes relacionadas con el transporte. Permite a los usuarios el acceso sin trabas a redes y a proveedores de servicios y/o servicios de su elección. Se soporta movilidad generalizada que permitirá la prestación coherente y ubicua de servicios a los usuarios".

A continuación se mencionan algunos de los aspectos, que según ITU [Itu04], caracterizan a las Redes de Próxima Generación:

- Transferencia basada en paquetes.
- Separación de las funciones de control en capacidades de portador, llamada/sesión, y aplicación/servicio.
- Separación entre la prestación del servicio y el transporte, y la provisión de interfaces abiertas.
- Capacidades de banda ancha con QoS extremo a extremo.
- Inter-funcionamiento con redes tradicionales a través de interfaces abiertas.
- Movilidad generalizada.
- Acceso sin restricciones de los usuarios a diferentes proveedores de servicios.
- Convergencia de servicios entre fijo y móvil.
- Independencia de las funciones relativas al servicio con respecto a las tecnologías de transporte subyacentes.

Es importante señalar que las Redes de Próxima Generación en ningún caso pretenden ser un reemplazo total de las redes ya existentes. La finalidad es hacer uso de las redes convencionales integrándolas con nuevas tecnologías, para que formen parte y sean la base de las NGN, con el objetivo de mantener las inversiones ya realizadas por parte de los operadores de telecomunicaciones.

2.1.3 Tendencias

Los operadores de telecomunicaciones desean más clientes con la finalidad de incrementar sus ingresos, necesitan reducir costos y poner en el mercado nuevos e innovadores servicios de la forma más rápida posible. Por otra parte, los usuarios desean pagar por nuevos servicios con calidad, simplicidad y personalización, movilidad y libertad dada por un acceso a la información donde sea y por el medio que sea. Todo lo anterior está derivando a que ocurra una gran transformación en las telecomunicaciones: La convergencia de redes fijas-móviles.

Algunos cambios que se han ido presentando en el tiempo en distintos ámbitos de las telecomunicaciones [Wil06] son las siguientes:

- Redes fijas: Los operadores de redes fijas enfrentan nuevos retos frente a operadores de bajo costo, operadores móviles, y a la tecnología VoIP. Aunque las ganancias sobre redes fijas de voz y datos son altas, se presenta una abrupta tendencia a la baja en el volumen de tráfico, a causa de nuevas tecnologías competitivas así como también una presión sobre los precios finales de los servicios. Surge una disminución en el uso de servicios clásicos.
- Redes móviles: hacia fines del 2005, se registraron cerca de dos billones de subscriptores móviles en el mundo, destacando un fuerte crecimiento en países en desarrollo (incluyendo Asia y Latinoamérica, y específicamente en Chile terminando el año 2007 se registró un 84% de usuarios móviles del total de la población [Sub07]). El servicio de voz es aún el principal recurso de ingresos y utilidades (cerca del 80% de los ingresos totales). Se presentan nuevas oportunidades de ganancias en servicios mejorados tales como *ring tones*, imágenes (*wallpapers*) y juegos, y en nuevos servicios de conferencia, *streaming*, Chat. Los operadores móviles comenzaron a abrir nuevos mercados globales, arrebatando minutos a las redes fijas, como también desarrollando nuevos servicios.
- Banda Ancha de Internet: a medida que esta tecnología prolifera, paralelamente el ancho de banda se ve incrementado. Sin embargo, las utilidades tienden a declinar a causa de la competitividad entre los proveedores.

En el siguiente gráfico podemos observar que la cantidad de usuarios de redes fijas casi se ha mantenido a través de los años en comparación a un rápido crecimiento en la cantidad de usuarios de Internet y de redes móviles.

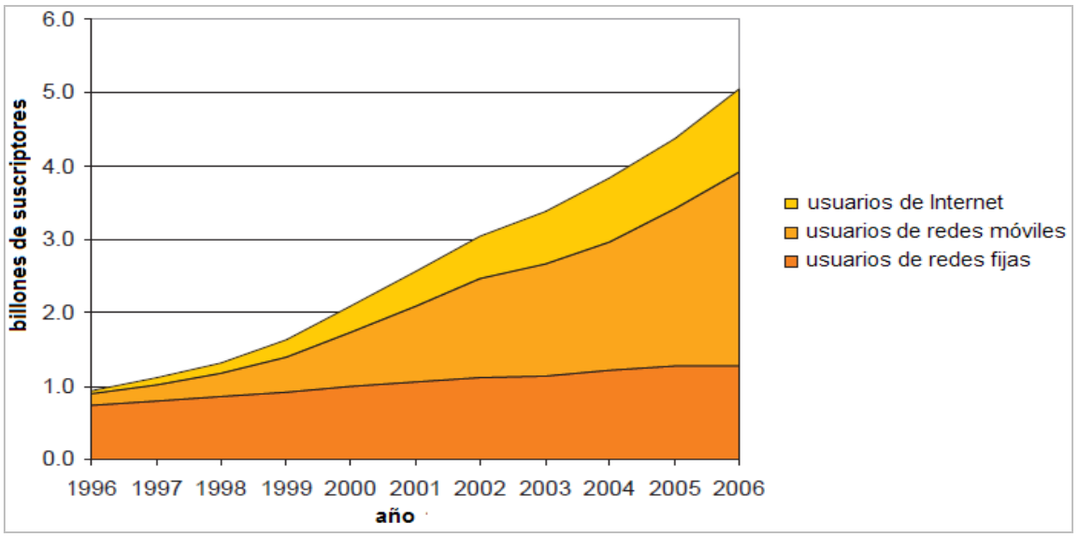


Figura 4. Crecimiento en redes fijas, móviles, e Internet, en cuanto a la cantidad de suscriptores (en billones entre los años 1996 y 2006).[9]

Tradicionalmente, el termino “Convergencia de redes fijas-móviles” ha sido usado por la industria de las telecomunicaciones cuando se habla de una integración entre tecnologías fijas e inalámbricas. Pero este no es el único punto de vista cuando se habla de convergencia, se puede hablar también de convergencia entre la industria de la información, la de contenidos multimedia y la industria de las telecomunicaciones [Eri05], o más en general, como la convergencia entre las tecnologías de la información y de las telecomunicaciones. Lo anterior, queda representado mediante la figura 5:

[9] Fuente: [Itu07]

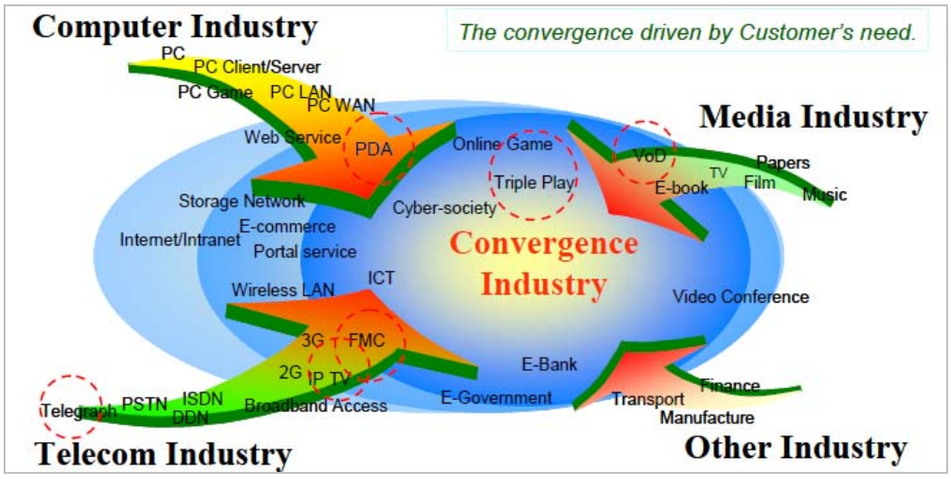


Figura 5. Convergencia: transformación de la industria.[10]

Es posible también mirar la convergencia desde otros puntos de vista: una convergencia de servicios, en donde, una multitud de servicios (persona a persona, persona a contenido y contenido a persona) pueden ser entregados bajo diferentes redes de acceso y a través de diferentes dispositivos; o también una convergencia de dispositivos, en donde dispositivos comunes soportan diferentes tipos de accesos a redes fijas o móviles (o inalámbricas) y poseen múltiples funcionalidades tales como TV/Video, e-mail, cámaras, entre otras [Eri05].

La figura 6 muestra tendencias en cuanto a la evolución de las redes de telecomunicaciones camino a una red global IP:

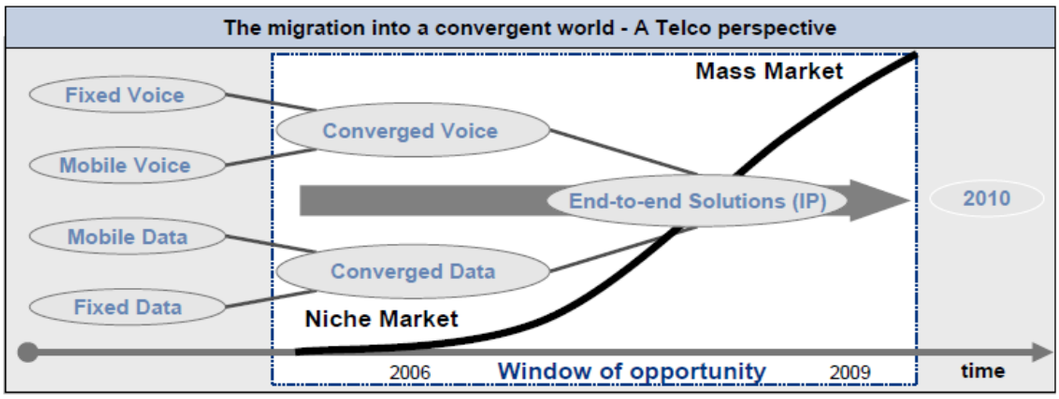


Figura 6. Tendencias en la convergencia de redes de telecomunicaciones.[11]

[10] Fuente: [Hua08]
[11] Fuente: [Det07]

En la figura 6 podemos observar que se pronostica que a fines del 2010 la mayoría de los operadores de telecomunicaciones habrán concluido la migración hacia las Redes de Próxima Generación.

Otra forma de graficar la evolución de las redes de telecomunicaciones, según el medio de acceso a éstas, es la que se muestra en la figura 7:

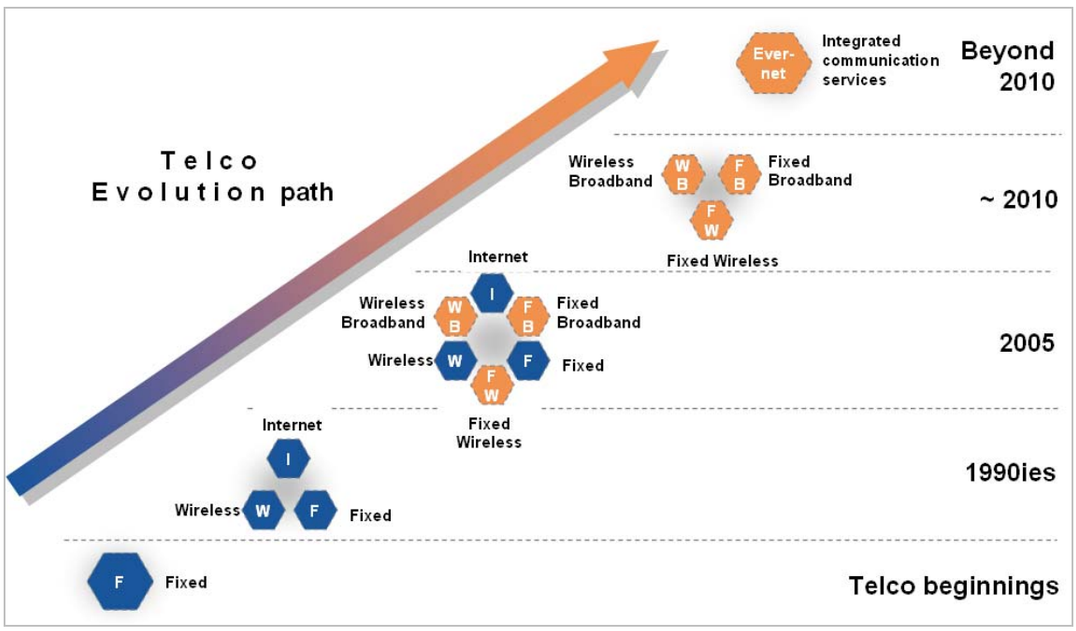


Figura 7. Evolución de las telecomunicaciones. [12]

2.1.4 Tecnologías Habilitantes para la convergencia

Términos claves en la convergencia de redes son IMS, SIP, Parlay/OSA y Parlay-X, tecnologías que posibilitan la creación de innovadores servicios, proporcionados a través de cualquier medio de acceso y utilizando cualquier dispositivo. A continuación se definirá cada uno de estos términos, y se describirá el papel que juegan en las Redes de Próxima Generación.

- **IMS** (*IP Multimedia Subsystem*): IMS es el primer paso para una red "*All-IP*" combinando lo mejor de dos mundos: la calidad e interoperabilidad de las telecomunicaciones y el rápido e innovador desarrollo de las tecnologías de la información dando énfasis principalmente al mundo de la Internet.

[12] Fuente: [Det07]

IMS es un conjunto de especificaciones que describen la arquitectura de las Redes de Próxima Generación para implementar telefonía basada en el protocolo IP y servicios multimedia. IMS define una completa arquitectura y marco de trabajo que posibilita la convergencia de voz (VoIP), video, datos y tecnología de las redes móviles, todo sobre una infraestructura basada en el protocolo IP [Ibm06]. La principal misión de IMS es proveer de Internet a cualquier dispositivo conectado a una red móvil.

IMS fue planeado para actuar como una arquitectura unificadora, capaz de soportar una gran variedad de servicios a través de redes basadas en conmutación de circuitos (*circuit-switches*) y en conmutación de paquetes (*packet-switches*), usando una variedad de diferentes tecnologías de acceso fijas e inalámbricas [Par04a].

Sin duda, IMS puede ser considerado como el corazón de las Redes de Próxima Generación, y su desarrollo es fruto del continuo esfuerzo de organizaciones internacionales como la 3GPP, que en colaboración con la ETSI definieron este conjunto de especificaciones.

- **SIP** (*Session Initiation Protocol*): SIP es el principal protocolo de señalización usado en redes IMS. Este fue desarrollado por la IETF y fue seleccionado por la 3GPP como un estándar para IMS. La función de SIP es establecer. Modificar y terminar sesiones multimedia de voz, video y Chat, bajo redes IP. Este protocolo de inicio de sesiones trabaja de extremo a extremo soportando el establecimiento y finalización de servicios como por ejemplo, ubicación de usuarios, disponibilidad de usuarios, capacidades del usuario, además de administración y configuración de sesiones. SIP también fue diseñado para gestionar sesiones multimedia adicionales, y los participantes involucrados, los que pueden ser agregados o removidos dinámicamente desde una sesión [Eri07]. Por todo lo anterior, SIP fue seleccionado como protocolo principal en la arquitectura de las Redes de Próxima Generación, además de presentar éste gran flexibilidad y seguridad.

- **Parlay/OSA y Parlay-X**: APIs estandarizadas y abiertas, desarrolladas por el foro Parlay, en cooperación con 3GPP y ETSI, que juegan un rol en la capa de servicios de la nueva arquitectura de redes convergentes. La principal característica buscada es que estos servicios pueden ser accedidos no sólo por los

operadores de las redes de telecomunicaciones, sino que también por terceras partes, las que pueden hacer uso de las capacidades y recursos de la red creando nuevas aplicaciones de forma segura y controlada [Par04b], de ahí la sigla "OSA" que significa "Arquitectura de Servicios Abiertos".

IMS, SIP y Parlay/OSA junto a Parlay-X se unen bajo una misma arquitectura para integrar las Redes de Próxima Generación, dando soporte a un amplio rango de servicios. Como se puede observar en la figura 8, esta arquitectura entrega la capacidad para desarrollar servicios multimedia que pueden ser accedidos por un usuario desde diversos dispositivos terminales a través de una red IP o una red tradicional de telefonía.

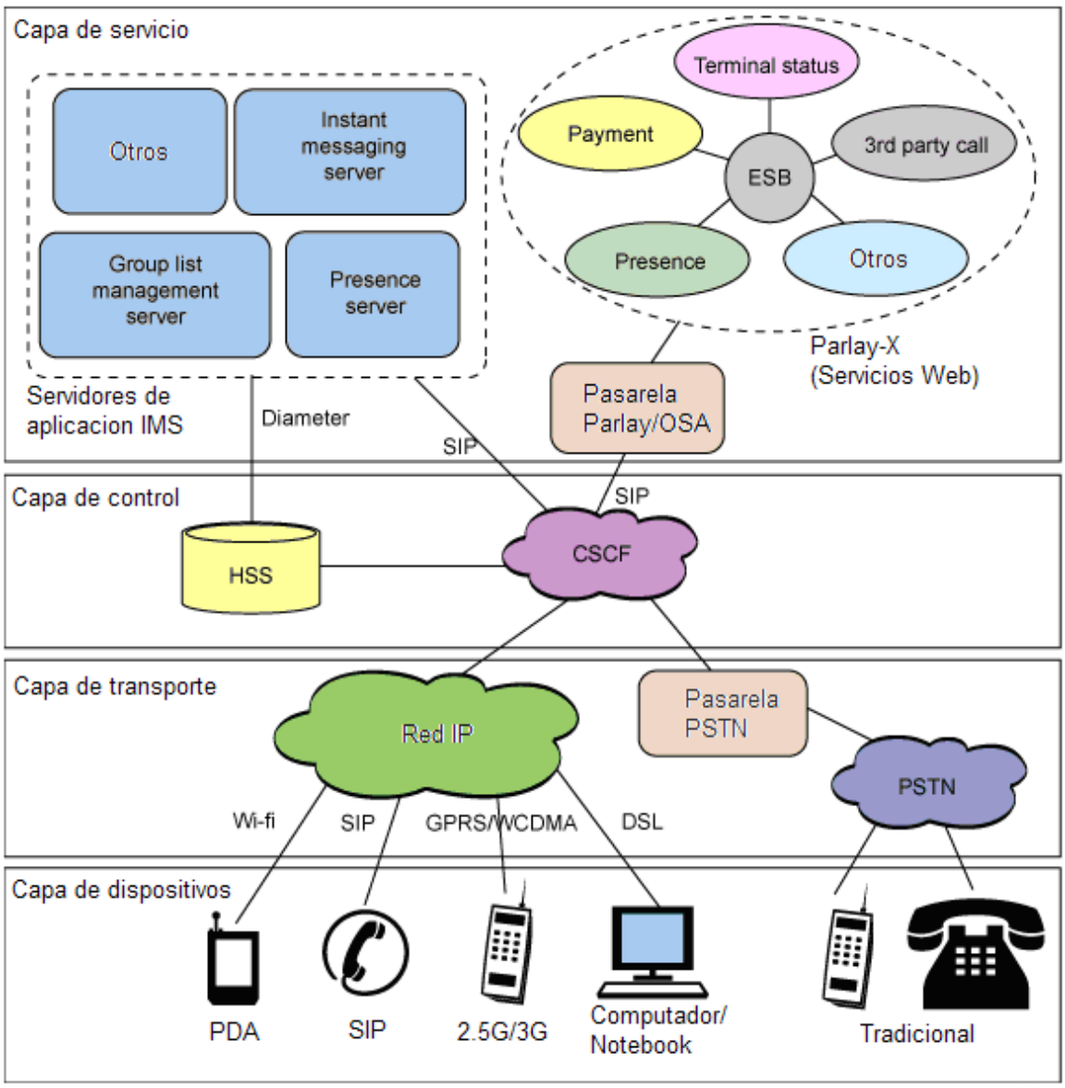


Figura 8. SIP, IMS y Parlay/OSA integrados bajo una misma arquitectura separada en capas.[13]

[13] Fuente: traducción al español del gráfico referenciado en [Ibm06].

Esta arquitectura de redes convergentes se divide en 4 capas o niveles principales:

- **Nivel de servicio o aplicación:** Es la capa superior en la arquitectura NGN. Las tres capas inferiores proveen una plataforma de redes estandarizada e integrada para permitir la entrega de una variedad se servicios en la capa superior. Parlay/OSA y Parlay-X juegan rol en éste nivel, en donde el componente más cercano a la capa de control es la pasarela Parlay/OSA (*Gateway* Parlay/OSA) mediante la cual las aplicaciones pueden hacer uso de las funcionalidades de las redes subyacentes.
- **Nivel de sesión o control:** La función de control de sesiones de llamadas (CSCF), término genérico asignado a servidores SIP, es uno de los elementos centrales de la capa de control, y cumple la función principal de crear y gestionar sesiones o llamadas, junto con registrar las terminaciones (*Endpoints*) y enrutamiento de los mensajes de señalización SIP hacia la capa superior de servicio.
- **Nivel de acceso o transporte:** Esta capa es responsable de iniciar y terminar las sesiones SIP y de proveer una conversión de datos transmitidos entre formatos análogo/digital y formato de paquetes IP. Además, la capa de transporte permite a los dispositivos crear y recibir llamadas hacia y desde redes PSTN de telefonía tradicional u otras redes de conmutación de circuitos a través de una pasarela (*Gateway*) PSTN.
- **Nivel de dispositivos de acceso:** La arquitectura NGN entrega la opción a los usuarios finales para que estos puedan acceder a la red mediante diversos dispositivos terminales, tales como computadores, celulares, PDAs, teléfonos digitales, etc. Otros dispositivos tales como teléfonos análogos tradicionales, no están capacitados para conectar directamente a la red IP, alternativamente, estos dispositivos establecen conexión a través de la pasarela PSTN.

Otra forma de representar la arquitectura anteriormente descrita es desmenuzándola con la finalidad de mostrar con mayor detalle los elementos presentes en cada capa. De esta forma, es posible destacar qué componentes integran a IMS, lo que queda graficado en la próxima figura.

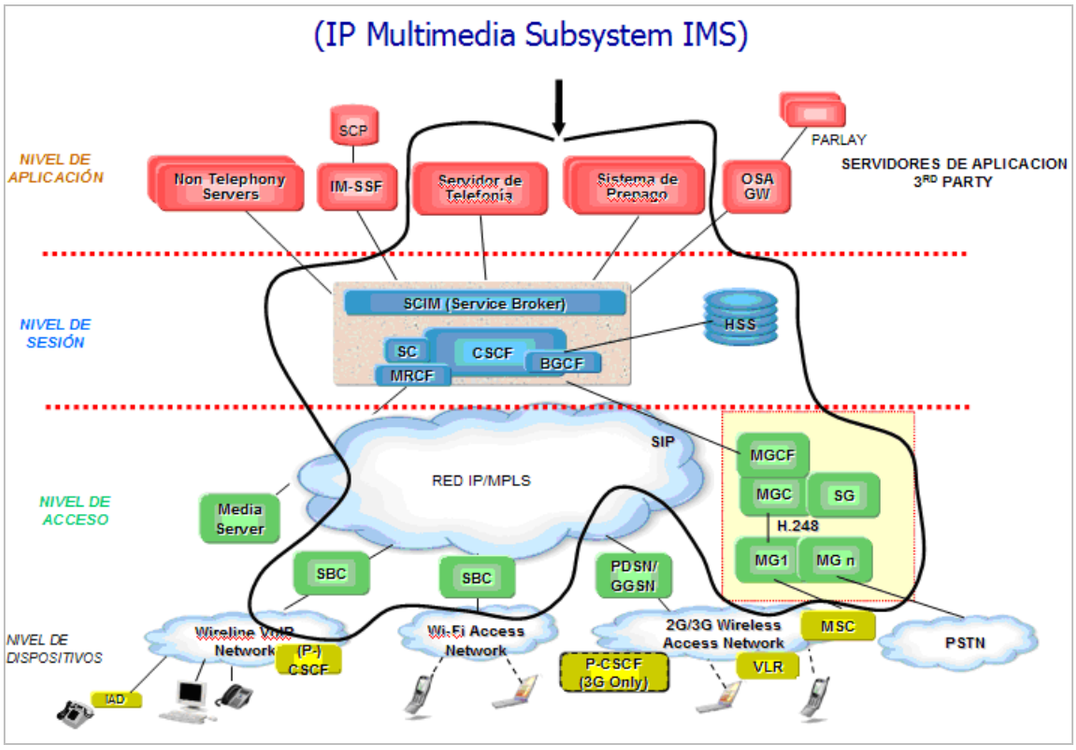


Figura 9. SIP, Componentes IMS y Parlay/OSA integrados bajo una misma arquitectura. [14]

2.2 Parlay/OSA y Parlay-X: Un nuevo enfoque en los negocios

2.2.1 Nuevos modelos de negocio: abriendo la red a terceros

Las nuevas necesidades de comunicación por parte de los usuarios están expandiendo las oportunidades de negocios para los operadores de telecomunicaciones, los proveedores de servicios y las empresas. Parlay/OSA está protagonizando un incremento masivo en la disponibilidad de servicios para los clientes, gracias a que este estándar abierto posibilita, de forma segura, la integración de las capacidades de las redes de telecomunicaciones con las tecnologías de la información, entregando asimismo, una infraestructura técnica que genera nuevas visiones de negocios para los operadores, abriendo nuevos canales, para que terceras partes desarrollen servicios y provean de contenidos a los usuarios finales. Un nuevo jugador ingresa, de esta forma, en el negocio de las telecomunicaciones, compitiendo en el mercado de los servicios, y lo más interesante, no necesitan ser operadores de una red [Aba02]. Algunos pueden

[14] Fuente: [Ber06].

entregar directamente servicios al usuario mientras el operador brinda conectividad a estos, y otros pueden llegar a los usuarios a través del operador. Esto se puede representar en la figura 10:

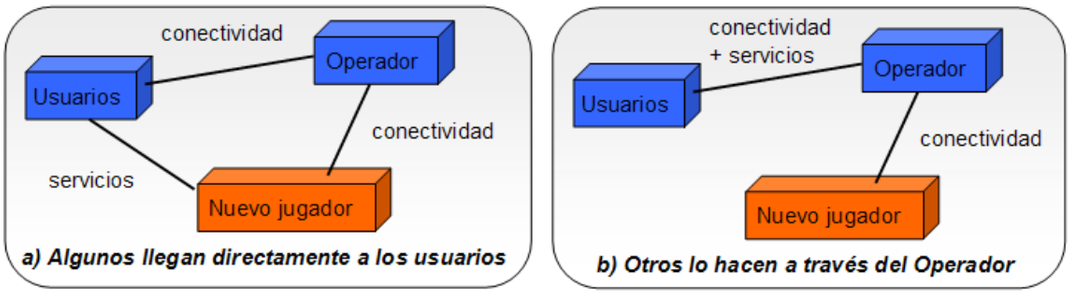


Figura 10. Parlay/OSA y los nuevos modelos de negocio. Un nuevo jugador entra el mercado de los servicios.

A continuación se describen algunos de los posibles nuevos jugadores [Aba02] que pueden entrar al nuevo entorno de negocio Parlay/OSA:

- MVNO: Operador móvil virtual (en inglés, *Mobile Virtual Network Operador*). Es una compañía de telefonía móvil que carece de una red propia. Para dar servicio, recurre a la cobertura de red de otra empresa que sí posee una red propia con la que subscribe un acuerdo.
- VNO: Operador de red virtual (en inglés, *Virtual Network Operador*). Es una compañía que provee de la administración de servicios y revende servicios de telecomunicaciones a otras empresas que poseen red.
- ASP: Proveedor de Servicios de Aplicaciones (en inglés, *Application Service Provider*). Es una empresa que entrega servicios de software a través de una red a múltiples entidades desde un centro de cómputo.

A pesar de que el costo de un nuevo servicio representa sólo una fracción del costo de la infraestructura total de la red, la capa de servicio conlleva potencialmente hacia un aumento significativo en el "promedio de ingresos por usuario" o ARPU por sus siglas en inglés (*Average Revenue Per User*) y expande la base de usuarios en nuevos mercados. Para los operadores, esto significa grandes retornos sobre las inversiones globales de la red no sólo mediante los ingresos obtenidos desde los servicios entregados actualmente, sino que contribuyendo también en las siguientes direcciones [Par05a]:

- Expandiendo la cadena de valor del operador, ingresando en nuevos segmentos de negocio y en nuevos mercados, por ejemplo, haciendo posible que los usuarios paguen su estacionamiento a través de sus teléfonos móviles.
- Expandiendo la marca e imagen de las compañías de telecomunicaciones mediante nuevas estrategias de mercado por medio del desarrollo de servicios de forma rápida y con bajo costo, a través del aprovechamiento de aplicaciones de terceras partes y el uso de proveedores externos de servicios de aplicaciones.
- Reduciendo el *time-to-market* o tiempo de puesta en el mercado de nuevos servicios para el aprovechamiento de las nuevas tendencias y enfocándose en pequeños y grandes nichos de mercado.
- Obteniendo nuevos ingresos desde servicios tradicionales integrándolos dentro de nuevos servicios con características mejoradas, por ejemplo, tendiendo un servicio de envío de email por voz sobre SMS con un número de remitente.
- Reduciendo la migración de clientes para los operadores, diferenciándose de la competencia, y ofreciendo servicios a los usuarios finales que pueden ser ampliamente personalizados.

2.2.2 Las ventajas de Parlay/OSA y Parlay-X

Con la introducción de la tecnología ofrecida por Parlay en las empresas de telecomunicaciones se produce una situación en donde todos ganan [Par05a]:

Para los operadores de redes:

- Operadores de redes tradicionales y virtuales (MVNO, VNO) entran en el nuevo entorno de negocio.
- Solución pensada con visión de futuro.
- Rápida obtención de ingresos.
- Solución independiente de la red subyacente sobre la cual se implementa este estándar.
- Independencia con la plataforma e interoperabilidad con el vendedor.
- Interoperabilidad con las aplicaciones desarrolladas.

- Evolución de servicios basados en Redes Inteligentes y en el protocolo IP.
- Fácil migración de servicios desde redes 2G, 2.5G a redes 3G.
- Arquitectura abierta segura.
- Reducción de costos y riesgos para expandir la oferta de negocios.

Para los proveedores de servicios:

- Solución pensada con visión de futuro.
- Rápida obtención de ingresos.
- Solución independiente de la red subyacente sobre la cual se implementa.
- Permite desarrollo de servicios por terceras partes.
- Interoperabilidad con las aplicaciones.
- Reducción de los costos y riesgos de desarrollar nuevos servicios.

Para los desarrolladores de software: Parlay provee un mercado global para terceras aplicaciones, y reduce los costos de integrar aplicaciones con la infraestructura de los proveedores de servicios. Otras ventajas para los desarrolladores de software son:

- Soluciones independientes de la red.
- Plataformas estándares.
- Rápido desarrollo de nuevos servicios, sobre todo los basados en Servicios Web.
- Plataforma para la ejecución de servicios de forma independiente.
- Herramientas estándares de desarrollo.
- Reducción de la necesidad de conocimiento especializado en telecomunicaciones.
- Amplia comunidad de desarrollo.
- Reducción de los costos y riegos de desarrollar nuevos servicios.

Gracias a las APIs entregadas por Parlay-X, mediante las cuales las capacidades de la red pueden ser accedidas y utilizadas a través de Servicios Web, el número de desarrolladores se incrementa considerablemente. Esto queda graficado en la figura 11:

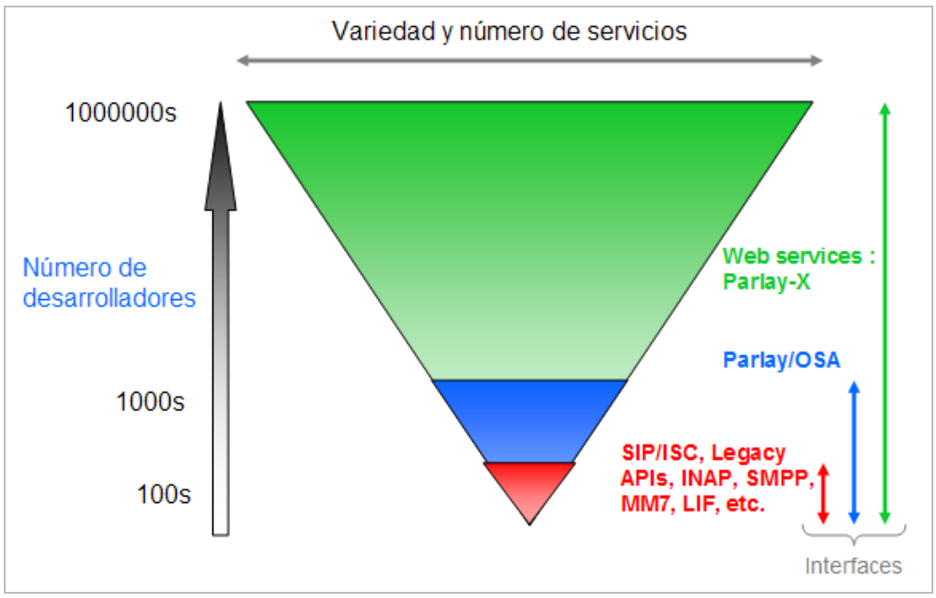


Figura 11. Cantidad de desarrolladores según la interfaz utilizada para acceder a las funcionalidades de una red de telecomunicaciones. En el gráfico, 100s, 1000s, 1000000s indica, cientos, miles, millones respectivamente.

En esta figura podemos observar que la disponibilidad de desarrolladores aumenta considerablemente a millones cuando se utilizan las APIs de Parlay-X para crear nuevos servicios de telecomunicaciones.

Para los integradores de sistemas: Parlay entrega un punto de integración estándar y abierto para el mundo de las aplicaciones, servicios, proveedores de contenidos, y para el mundo de los operadores de redes de telecomunicaciones y de los habilitadores de capacidades de servicios de telecomunicaciones. Esto reduce el costo de integrar aplicaciones con la infraestructura de los proveedores de servicios y de agregar nuevas capacidades a la infraestructura. Otras ventajas para los integradores de sistemas son:

- Reducción de costos y riesgos al proveer un simple punto de integración de servicios, contenido y habilitadores de redes.
- Soluciones independientes de la red de telecomunicaciones.
- Plataformas estándares.
- Plataforma para la ejecución de servicios de forma independiente.

- Herramientas estándares de desarrollo.

Para los usuarios finales: Parlay trae beneficios para los usuarios finales mayormente por la expansión masiva de la disponibilidad de servicio, la posibilidad de que los operadores actúen como un minorista de servicios para el usuario, y por el incremento de la interoperabilidad de aplicaciones entre los distintos tipos de redes. Estos beneficios también incluyen:

- Un amplio rango de servicios.
- Atractiva oferta de servicios de calidad.
- Portafolio personalizado de servicios con oferta enfocada.
- Conveniencia y eficiencia.

CAPÍTULO III: EL ESTÁNDAR

3.1 Estándar Parlay/OSA

3.1.1 Arquitectura conceptual

En el punto 2.1.4 se describieron las tecnologías habilitantes necesarias para llevar a cabo una migración hacia las Redes de Próxima Generación, y se indicó y graficó en qué capa de la arquitectura general de estas redes juega su rol Parlay/OSA, específicamente en la capa de servicio o aplicación. A continuación se describe la arquitectura interna de Parlay/OSA.

Parlay/OSA define una arquitectura *middleware* (capa intermedia de software) que posibilita la interacción entre aplicaciones informáticas y las capacidades de las redes de telecomunicaciones, a través de un conjunto de interfaces estandarizadas y abiertas denominadas APIs de Parlay/OSA. Cada funcionalidad de la red queda conceptualizada en las llamadas "Capacidades de servicio" o SCFs (en ingles, *Service Capabilty Features*). En el desarrollo de éste y de los próximos capítulos se hablará indistintamente de los términos SCF y "Capacidad de servicio", para referirse a una funcionalidad de la red de telecomunicaciones subyacente.

Las aplicaciones desarrolladas podrán hacer uso de manera integrada de las capacidades de la red subyacente mediante interfaces distribuidas especificadas por las APIs de las SCFs, las que a su vez, son agrupadas y conceptualizadas en los llamados "Servidores de Capacidades de Servicio" o SCSs (en ingles, *Service Capability Servers*) que son entidades lógicas que actúan como intermediarios entre las aplicaciones de los usuarios finales y los elementos de la red subyacente [Fal04]. Por otra parte, las funcionalidades de la red y la integridad de ésta son algo que debe ser cuidado y protegido. Por esta razón, la relación entre una aplicación cliente y un SCS debe ser administrado, y es aquí en donde aparece un tercer elemento: el *framework*, que junto con los SCSs, y las capacidades de servicios integradas en estos SCSs forman la llamada Pasarela o *Gateway* Parlay/OSA. Los elementos anteriormente señalados y sus interacciones son ilustrados a través de la figura 12:

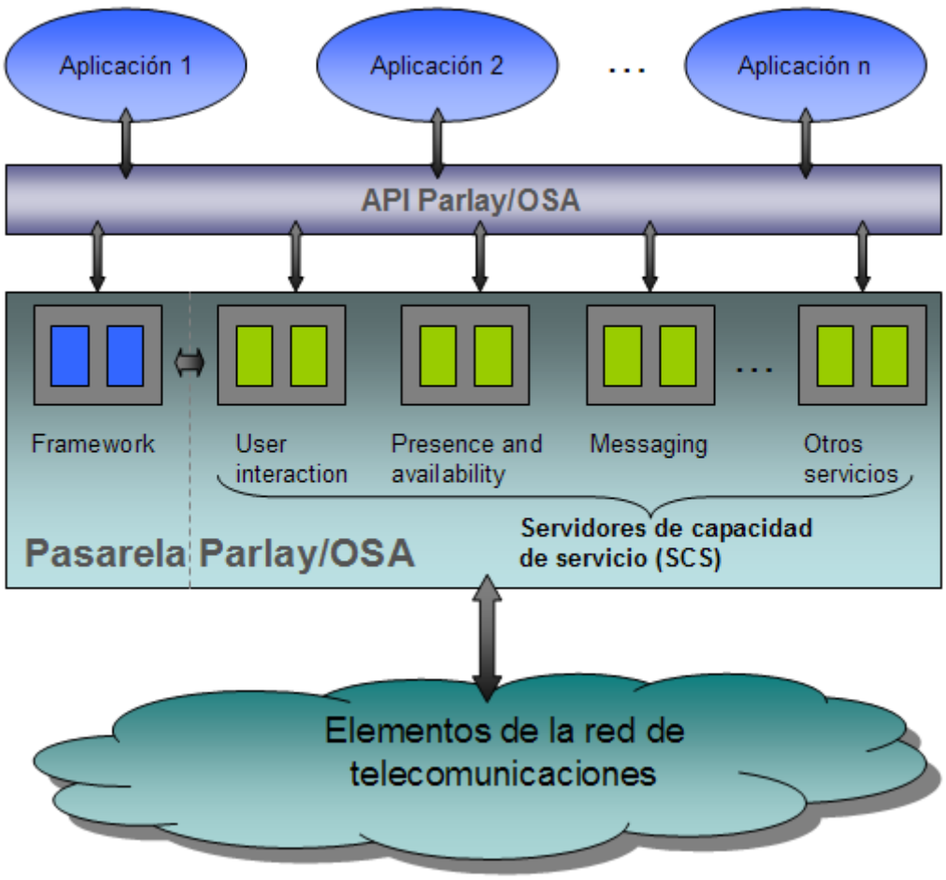


Figura 12. Arquitectura de Parlay/OSA

Como se puede ver en la figura anterior, la pasarela Parlay/OSA interactúa a través de su API con las aplicaciones cliente. Las interacciones entre el resto de los componentes de la pasarela, quedan especificadas a través de interfaces entre:

- El *framework* y las aplicaciones cliente.
- Los SCSs y la aplicación.
- Los SCSs y los mapeos hacia los elementos de la red subyacente.
- El *framework* y los SCSs.

Se observan entonces, tres grandes elementos en esta arquitectura: El *framework*, los SCSs (o grupo de SCFs) y las aplicaciones clientes. Estos elementos son detallados en los puntos 3.1.1.1, .3.1.1.2 y 3.1.1.3.

3.1.1.1 La aplicación cliente

El propósito de la aplicación cliente es proveer una interfaz humano-máquina para uno o más usuarios finales. Desde que los protocolos de red abren un interesante

camino para que los humanos se comuniquen, la aplicación debe asegurar que la petición desde un usuario sea traducida a una operación en la red de telecomunicación subyacente. Similarmente, se debe asegurar que un evento en la red sea traducido en una indicación para el usuario. Parlay/OSA oculta la complejidad de la red, así una aplicación cliente traduce las operaciones Parlay en eventos para los usuarios, y viceversa. Los SCSs o concretamente las SCFs que los integran, completan el proceso traduciendo las operaciones Parlay en eventos de redes [Unm06]. La figura 13 ilustra dicha interacción:

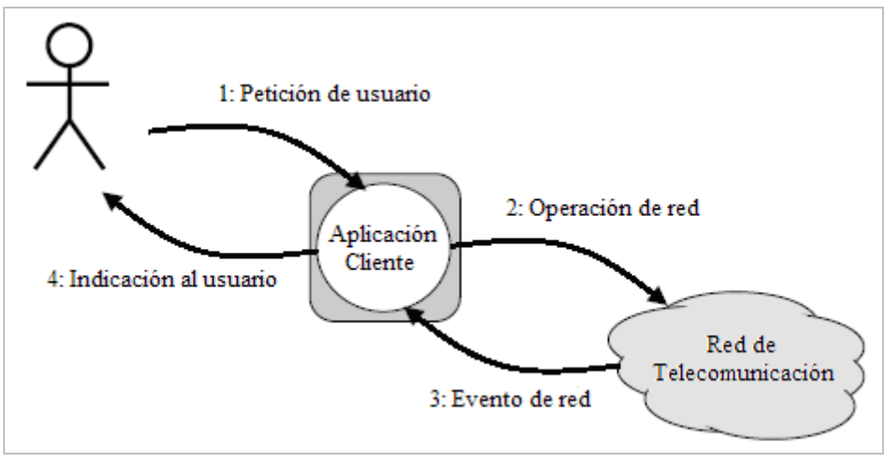


Figura 13. Aplicación cliente interactuando con la red, mediante una petición del usuario final.

La aplicación cliente provee así una interfaz a sus usuarios y hace uso de las interfaces entregadas por los SCSs para iniciar operaciones que se llevan a cabo en la red o para ser informadas de lo que pasa en la red. Las interfaces a los usuarios no son definidas por Parlay, las que pueden consistir en botones e interfaces gráficas sobre un teléfono o gráficos sobre una PDA, lo que escapa del alcance de Parlay/OSA. Las interfaces hacia las SCSs, sin embargo, son parte de Parlay. El rol de los SCSs es cubierto en el próximo punto.

3.1.1.2 Los SCSs o Servidores de Capacidades de Servicio.

Esencialmente los SCSs traducen las peticiones desde Parlay hacia el lenguaje de las redes subyacente, y viceversa. Así, los SCSs traducen peticiones (invocación de métodos) desde la aplicación cliente en operaciones en la red, entregando a la aplicación cliente una vista abstracta de las capacidades disponibles en la red. Entonces, la aplicación cliente sólo necesita entender como el SCS opera y no como opera la red. Incluso, el SCS podría interactuar con diferentes tipos de redes, todas entregando el

mismo tipo de función (por ejemplo, control de llamadas), pero para cada una usando una diferente interfaz o protocolo [Unm06]. Lo anterior se puede graficar en la figura 14:

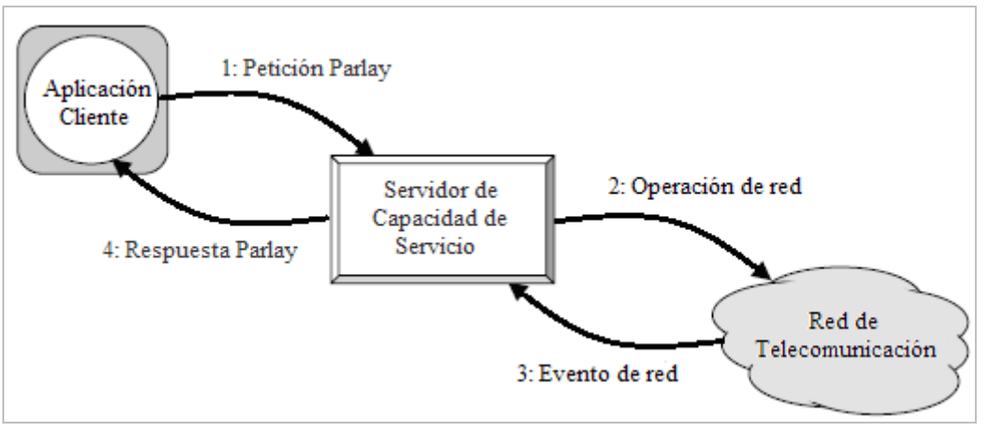


Figura 14. Servidor de Capacidad de Servicio interactuando con la red, a petición de la aplicación cliente.

3.1.1.3 El framework.

El *framework* autentica a la aplicación cliente para confirmar su identidad, y para asegurar que la aplicación cliente es quien dice ser. Uno de los objetivos de Parlay es que el *framework* y la aplicación cliente puedan estar en dominios diferentes de seguridad, así la aplicación cliente puede asegurarse de que está interactuando con un *framework* genuino. Por esta razón la autenticación es mutua y siempre es iniciada por la aplicación cliente. De esta misma manera, también existe un dominio de seguridad para los SCSs, con la finalidad de que una autenticación mutua pueda también ser aplicada entre el *framework* y los SCSs [Unm06]. Por lo tanto, el *framework* provee de acceso controlado a los SCSs, y por ende, a las SCFs o capacidades de servicios.

El *framework* de Parlay/OSA consiste en una familia de interfaces, de las cuales, se nombran a continuación las más importantes:

- Administración segura y de confianza: autenticación de dominios.
- Registro de las SCFs o capacidades de servicio: registro de nuevos SCFs en el *framework.*
- SCF *Factory*: creación de nuevas instancias de una SCF.
- SCF *Discovery*: descubrimiento de las SCFs ofrecidas por el operador e implementadas en la pasarela Parlay/OSA.

3.1.1.4 Interacción entre los tres componentes.

Para que las aplicaciones cliente y las capacidades de servicio se comuniquen se requiere de que el *framework* actúe como intermediario garantizando la seguridad del sistema. En primer lugar, todas los SCSs deberán registrar las capacidades de servicios o SCFs en el *framework.* Cuando una aplicación desee usar una funcionalidad determinada de la red, ésta deberá interactuar con el *framework* autenticándose y solicitando acceso a la capacidad de servicio que utilizará. El *framework*, tras autenticar y autorizar a la aplicación el acceso a la red, usará la SCF registrada de acuerdo a la capacidad requerida, para crear una nueva instancia de dicha SCF proporcionando la interfaz a la aplicación cliente.

A continuación se describe de forma detallada cada uno de los pasos mencionados anteriormente, los que serán explicados con el apoyo de la figura 15:

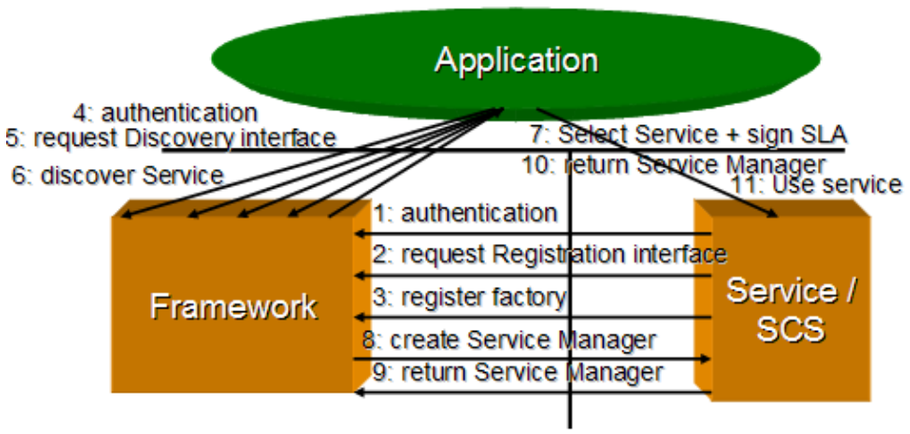


Figura 15. Registro y descubrimiento de servicios (SCFs), y uso de éstos por parte de aplicaciones clientes.[15]

Basándose en el grafico anterior, tres diferentes etapas se pueden distinguir en el proceso de utilización de las capacidades entregadas por la red por parte de las aplicaciones cliente [Moe02]:

1. Registro: En esta etapa una nueva capacidad de servicio o SCF es agregada. Primero el servicio se comunica con el *framework* y pide a éste una interfaz de registro. El *framework* retorna una referencia de esta interfaz. Estos corresponden a los pasos 1 y 2 en la figura mostrada anteriormente. Luego, el

[15] Fuente: [Moe02]

SCS solicitado usa esta interfaz para publicar su SCF, por ejemplo, la SCF de "Control de llamadas de multi-partes" (en inglés, *Multi-Party Call Control*) junto con sus capacidades, entre las cuales una podría ser, qué cantidad de partes por llamada es capaz de manejar. Posteriormente, el SCS entregará al *framework* su propia referencia a través de la interfaz de registro (paso 3 en la figura anterior). En este punto, el *framework* y el SCS conocen quien es quien.

2. Establecimiento de los acuerdos del servicio: En esta fase, se establecen las condiciones sobre las cuales se permite a la aplicación hacer uso del SCS. El operador del *framework* es capaz de ingresar las condiciones de uso de los SCSs. Una de esas condiciones, por ejemplo, es que la aplicación cliente sólo está permitida para usar un máximo de 4 partes por llamada.

3. Establecimiento de la comunicación en tiempo de ejecución: En esta etapa, se establece la comunicación entre la aplicación y el SCS. El paso final ocurre cuando la aplicación se comunica con el *framework* y usa la "interfaz de descubrimiento" para buscar qué SCFs están soportados en la pasarela Parlay/OSA (pasos 4 al 6). Continuado con el ejemplo del control de llamadas multi-parte, la aplicación pregunta y solicita al *framework*, a través de la "interfaz de descubrimiento", el retorno de todas las implementaciones de SCFs que cumplen con la petición de cuatro partes por llamada. El *framework* entonces verifica el "acuerdo de servicio" para encontrar si la aplicación está efectivamente permitida para usar el SCF "control de llamadas multi-partes". Al permitirla, se retornará una lista de implementaciones del SCF que sea capaz de manejar un máximo de 4 partes por llamada, en donde la aplicación usará una de ellas (paso 7). Luego, el *framework* solicitará el SCS específico (vía la interfaz "*Service Factory*") para crear una instancia del SCF que será usada por la aplicación. El *framework* también enviará la información sobre las condiciones bajo las cuales la aplicación puede hacer uso del SCF, por ejemplo, el máximo de cuatro partes permitidas por llamada (paso 8). El SCS creará la instancia del SCF correspondiente y retornará una referencia al *framework* (paso 9). El *framework* devolverá entonces esta referencia a la aplicación (paso 10). Desde ese momento, la aplicación queda capacitada para hacer uso del SCF requerido.

Es importante señalar que Parlay/OSA no sólo se enfoca en aplicaciones de terceras partes, sino que también es posible integrar aplicaciones y nuevas SCSs en el

mismo dominio del operador dueño de la pasarela Parlay/OSA. En este caso, como se trata de un dominio seguro, la autenticación (pasos 1 al 4) no es necesaria.

3.1.2 Servicios disponibles en el gateway Parlay/OSA 5.0

En el presente proyecto se toma como referencia la versión 5.0 de Parlay/OSA, usada también en la implementación del servicio desarrollado, descrita en el capítulo 5, junto con el *framework* utilizado e implementado bajo la misma versión del estándar.

A continuación se describen las Capacidades de Servicios o SCFs disponibles para ser implementadas en la especificación Parlay/OSA 5.0 [Ets05a]:

- Control genérico de llamadas: éste es un conjunto de procedimientos interactivos requerido para establecer, mantener y realizar llamadas entre dos usuarios, ya sea mediante una comunicación básica o una comunicación multimedia entre estos.
- Control de llamadas entre múltiples partes: esta capacidad de servicio es una mejora de las funcionalidades del control genérico de llamadas permitiendo establecer llamadas entre múltiples partes.
- Control de llamadas multimedia: consiste en una mejora de las funcionalidades del control de llamadas entre múltiples partes con capacidades multimediales, permitiendo un control granular de un *streaming* multimedia.
- Interacción con el usuario: funciones de capacidad de servicios para obtener información de los usuarios finales, reproducir anuncios o enviar mensajes cortos.
- Movilidad: funciones que permite saber la ubicación geográfica de un usuario, ya sea obteniendo la información a través de las celdas en caso que se establezca una comunicación móvil, mediante dispositivos GPS, o a través terminales fijos PSTN o IP.
- Capacidades del terminal: provee de procedimientos para obtener las capacidades del terminal que utiliza el usuario para acceder a la red.
- Control de sesión de datos: permite a las aplicaciones manejar las sesiones de datos iniciadas desde los terminales de usuario. Se aplica típicamente para redes GPRS.

- Mensajería genérica: permite acceder a casillas de correo electrónico, y enviar y recuperar mensajes, y controlar la interacción de las aplicaciones con los sistemas de mensajería de voz, FAX o email.
- Mensajería multimedia: es usada por aplicaciones para enviar, almacenar y recibir mensajes dentro de un contexto de casilla de correo, o fuera de él. También soporta mensajería en el contexto de la Mensajería Instantánea, SMS, MMS, GSM USSD, etc.
- Control de conectividad: provee de funcionalidades a las aplicaciones para que puedan controlar o influenciar la calidad de servicio entre dos puntos de la red.
- Control de cuentas: habilita a las aplicaciones para recuperar información de cuentas de usuario e historiales de transacciones.
- Gestión de políticas: permite la definición de políticas a aplicar a los servicios de usuario.
- Manejo de presencia y disponibilidad: provee de funcionalidades a las aplicaciones para que éstas obtengan y e ingresen información sobre la presencia de los usuarios y su disponibilidad. Por ejemplo, información de presencia podría ser: “en casa”, “estoy frente al computador”, “ausente”, etc.

3.1.3 Especificaciones de las APIs de Parlay/OSA

El principal objetivo de Parlay/OSA es identificar y especificar interfaces programables de redes con la finalidad de crear fácilmente aplicaciones usando los servicios entregados por las redes de telecomunicaciones. El set de SCFs o “Capacidades de servicios” (descritas en el punto 3.1.2) puede ser incrementalmente extendido, debido a que Parlay/OSA apunta a la especificación de interfaces extendibles y escalables que permitan la inclusión de nuevas funcionalidades en la red en futuras liberaciones con un mínimo impacto en las aplicaciones ya implementadas.

3.1.4 Esquemas de versionamiento

Debido a que varias organizaciones internacionales de estandarización han ingresado a la definición de las APIs de Parlay/OSA en distintas etapas, el resultado en el número de versión para cada especificación se ve desalineado. En la tabla 1 se indican las distintas versiones liberadas por ETSI, Parlay y 3GPP:

Tabla 1. Desalineamiento en las versiones de Parlay para una misma especificación.

Versión ETSI	Versión Parlay	Versión 3GPP	Fecha Publicación
N/A	Parlay 2.1	3GPP OSA Release 99 (3G TS 29.198 v3.x.y)	Junio 2000
ETSI OSA Phase 1 (ES 201 915)	Parlay 3	3GPP OSA Release 4 (3G TS 29.198 v4.x.y)	Febrero 2002
ETSI OSA Phase 2 (ES 202 915)	Parlay 4	3GPP OSA Release 5 (3G TS 29.198 v5.x.y)	Enero 2003
ETSI OSA Phase 3 (ES 203 915)	Parlay 5	3GPP OSA Release 6 (3G TS 29.198 v6.x.y)	Abril 2005
ETSI OSA Phase 4 (ES 204 915)	Parlay 6	3GPP OSA Release 7 (3G TS 29.198 v7.x.y)	Abril 2007

En este proyecto se toman como referencia las especificaciones liberadas por ETSI para la versión 5.0 de Parlay/OSA (ES 203 915).

3.1.5 APIs de Parlay/OSA: diseño bajo UML

Con el propósito de que exista un trabajo armonizado y alineado al momento de definir los conjuntos de APIs, estás fueron diseñadas bajo modelos UML comunes, los que aseguran que todos los artefactos publicados estén bien alineados, desde un punto de vista técnico, lo que es necesario cuando trabajan varias organizaciones estandarizadoras en una misma especificación, como es el caso de Parlay/OSA. Para cada API de Parlay hay un modelo formal especificando la interfaz completa, su definición y su comportamiento. UML consiste en una colección de técnicas de modelamiento incluyendo diagramas de clases, diagramas de secuencias, y diagramas de transición de estados usados para modelar sistemas distribuidos de comunicación. Cada una de esas técnicas es usada en la definición de Parlay/OSA. Con lo anterior, se obtienen APIs que pueden ser implementadas finalmente sobre la tecnología mas adecuada. Estas APIs describen qué es lo que se tiene que hacer, y no el cómo se tiene que hacer.

Los documentos de especificación actuales son automáticamente generados desde este modelo UML común, usando herramientas de software. Esto asegura que las versiones de las especificaciones de ETSI y Parlay son semánticamente y sintácticamente exactas a las especificadas por la 3GPP. Parlay y ETSI publican de forma conjunta sus especificaciones, con el prefijo ES (*European Standards*), mientras que la 3GPP publica sus especificaciones bajo las siglas 3G TS (3GPP *Technical Specifications*) [Unm06].

Acompañando a cada documento de especificación, se adjuntan también interfaces descritas en tres tecnologías: IDL (CORBA), Java y WSDL.

3.1.5.1 Tecnología CORBA IDL

CORBA o Arquitectura Común de Intermediarios en Peticiones a Objetos (en inglés, *Common Object Request Broker Architecture*) es una solución *middleware* (capa intermedia de software) orientada a objetos para sistemas de comunicación basados en el modelo Cliente-Servidor [Omg]. Con el uso de tecnología CORBA, Parlay define los archivos IDL (lenguaje de definición de interfaz) para la especificación de las APIs.

En un principio, específicamente en la version 1.2 de Parlay/OSA, existían dos *middleware* disponibles: MIDL (*Microsoft Interface Definition Language*) y DCOM (*Distributed Component Object Model*). En ese entonces, los modelos UML de Parlay no estaban disponibles, ya que MIDL y DCOM no contribuían en una definición íntegra de las versiones de Parlay, fue por eso que CORBA IDL permaneció como única tecnología *middleware* de Parlay, además de presentar éste, requisitos de fiabilidad y prestaciones típicos de las redes de telecomunicaciones.

3.1.5.2 Tecnología Java™

Desde que los modelos UML de Parlay estuvieron disponibles, se abrieron alternativas de la incorporación de nuevas tecnologías distintas a CORBA e IDL. Dentro del grupo Parlay, el subgrupo de trabajo Java dirigió sus esfuerzos para producir tecnología que soportara tanto los modelos de programación de *Java2 Enterprise Edition* (J2EE™) como los de *Java2 Standard Edition* (J2SE™). Resultaron, de esta forma, equivalentes de IDL para J2SE, y para J2EE, definiendo en este último, una API local en los servidores de aplicaciones de Parlay.

3.1.5.3 Tecnología WSDL

La tercera tecnología utilizada por Parlay es la tecnología WSDL o Lenguaje de Definición de Servicios Web (en ingles, *Web Services Definition Language*), como consecuencia del creciente interés en tecnologías basadas en XML y del despliegue de los llamados Servicios Web.

Con las tres tecnologías nombradas anteriormente, todas derivadas desde un modelo UML común para Parlay, las organizaciones que deseen implementar una solución Parlay/OSA para su arquitectura de servicios, están libres de seleccionar una

tecnología particular que se ajuste de la mejor forma a sus sistemas, requerimientos específicos de implementación, o ambiente de desarrollo o destrezas de sus comunidades de desarrolladores.

3.1.6 Documentos de especificación para Parlay/OSA 5.0

A través del sitio oficial de ETSI (http://www.etsi.org), o alternativamente, del grupo Parlay (http://www.parlay.org), es posible obtener los documentos que especifican las APIs de Parlay/OSA. Se describirá la estructura de documentos para la versión 5.0 de Parlay/OSA.

La especificación de las APIs liberadas por ETSI (ES 203 915), correspondiente a la versión 5.0 de Parlay/OSA, se encuentran estructurada en las siguientes partes:

Tabla 2. Lista de documentos que especifican las APIs de Parlay/OSA versión 5.0

Título del documento	Versión de la especificación
Parte 1: Overview	ETSI ES 203 915-1 V1.1.1 (2005-04)
Parte 2: Common Data Definitions	ETSI ES 203 915-2 V1.1.1 (2005-04)
Parte 3: Framework	ETSI ES 203 915-3 V1.1.1 (2005-04)
Parte 4.1: Call Control Common Definitions	ETSI ES 203 915-4-1 V1.1.1 (2005-04)
Parte 4.2: Generic Call Control SCF	ETSI ES 203 915-4-2 V1.1.1 (2005-04)
Parte 4.3: Multi-Party Call Control SCF	ETSI ES 203 915-4-3 V1.1.1 (2005-04)
Parte 4.4: Multi-Media Call Control SCF	ETSI ES 203 915-4-4 V1.1.1 (2005-04)
Parte 4.5: Conference Call Control SCF	ETSI ES 203 915-4-5 V1.1.1 (2005-04)
Parte 5: User Interaction SCF	ETSI ES 203 915-5 V1.1.1 (2005-04)
Parte 6: Mobility SCF	ETSI ES 203 915-6 V1.1.1 (2005-04)
Parte 7: Terminal Capabilities SCF	ETSI ES 203 915-7 V1.1.1 (2005-04)
Parte 8: Data Session Control SCF	ETSI ES 203 915-8 V1.1.1 (2005-04)
Parte 9: Generic Messaging SCF	ETSI ES 203 915-9 V1.1.1 (2005-04)
Parte 10: Connectivity Manager SCF	ETSI ES 203 915-10 V1.1.1 (2005-04)
Parte 11: Account Management SCF	ETSI ES 203 915-11 V1.1.1 (2005-04)
Parte 12: Charging SCF	ETSI ES 203 915-12 V1.1.1 (2005-04)
Parte 13: Policy management SCF	ETSI ES 203 915-13 V1.1.1 (2005-04)
Parte 14: Presence and Availability Management SCF	ETSI ES 203 915-14 V1.1.1 (2005-04)

Además de la documentación para cada parte de la API, se encuentran disponibles los archivos IDL de CORBA, WSDL y las interfaces para J2SE y J2EE.

3.2 Parlay-X

3.2.1 Arquitectura conceptual

Uno de los objetivos de Parlay es incentivar a una gran comunidad de profesionales para que diseñen y construyan soluciones de valor agregado a los negocios sobre complejas redes de telecomunicaciones.

Parlay-X nació como una iniciativa para simplificar el modelo de Parlay/OSA, con la finalidad de que pueda ser usado con mayor facilidad y dinamismo por profesionales del mundo de la programación que no tengan que ser necesariamente técnicos en redes de telecomunicaciones.

El grupo de trabajo de Parlay-X fue creado en Septiembre del 2001 en respuesta a la creciente popularidad de las tecnologías SOAP y XML, las que unidas forman a WSDL, resultando ser un mecanismo alternativo de distribución distinto a la aproximación de CORBA/IDL [Unm06]. Gracias a que las APIs de Parlay-X pueden ser implementadas mediante Servicios Web usando WSDL, el modelo en el que se basan, entrega, respecto de Parlay/OSA, una mayor abstracción de las funcionalidades que ofrecen las redes subyacentes en torno a una serie de Capacidades de Servicios, término ya usado ampliamente en la descripción del estándar de Parlay/OSA.

Los Servicios Web de Parlay-X pueden ser implementados invocando las funcionalidades disponibles en una Pasarela o *Gateway* de Parlay/OSA mediante un adecuado mapeo entre Parlay-X y Parlay/OSA, o de manera alternativa, el servidor de Parlay-X puede implementar Servicios Web mediante una conexión directa con los elementos de la red de telecomunicaciones. Una Aplicación implementada bajo los estándares de Parlay-X puede ser escrita en cualquier lenguaje que soporte invocaciones a Servicios Web [Par05b]. Existen varias herramientas de desarrollo, para diferentes leguajes de programación, para crear, implementar e interactuar con *Web Services*.

La arquitectura de Parlay-X, su interacción con la pasarela de Parlay/OSA y con los elementos de la red, quedan ilustradas en la figura 16:

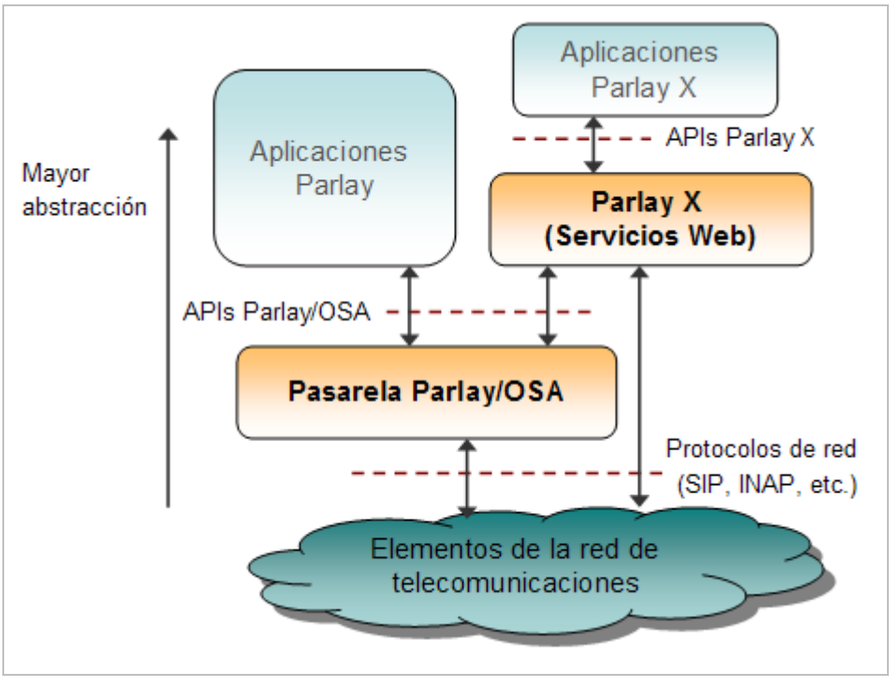


Figura 16. Arquitectura de Parlay-X y su relación con Parlay/OSA.

3.2.2 Características de Parlay-X

Los Servicios Web de Parlay-X presentan grandes beneficios: son relativamente simples, potentes, poseen un alto nivel de abstracción, abren la imaginación, para que desarrolladores y comunidades de las tecnologías de la información puedan comprender rápidamente las capacidades que entregan las redes de telecomunicaciones, construyendo nuevas e innovadoras aplicaciones. Se presentan, entonces, los siguientes rasgos:

- Cada Servicio Web es una abstracción de un conjunto de capacidades de telecomunicaciones, enfocándose más en la simplicidad que en la funcionalidad.
- La interacción entre una aplicación cliente y la pasarela de Parlay-X, en donde se implementan los Servicios Web especificados por las APIs de Parlay-X, se lleva a cabo con un intercambio basado en mensajes XML.
- Los Servicios Web de Parlay-X siguen una simple aplicación de semántica, permitiendo a los desarrolladores enfocarse en el acceso a las capacidades de la red subyacente, usando simples técnicas de programación de Servicios Web.
- Parlay-X es independiente de la red sobre la cual se implementa, en donde las Capacidades de Servicios son más relevantes que el tipo de red.

3.2.3 Capacidades de servicios en Parlay-X

La versión 2.1 de Parlay-X liberada por Parlay y ETSI en diciembre del 2006, define especificaciones de Servicios Web para las siguientes capacidades de servicio [Ets05b]:

- Llamadas por terceras partes: permite crear y administrar llamadas iniciadas por aplicaciones de terceros.
- Notificación de llamada: Gestiona llamadas generadas por los suscriptores en una red. Una aplicación de terceros determina como debe ser tratada la llamada. Es posible permitir a las aplicaciones direccionar la llamada, por ejemplo a un IVR (respuesta interactiva por voz), continuar la llamada o dar término a ésta.
- Mensajería corta: Provee la capacidad para enviar y recibir mensajes SMS de una manera simple. También es posible realizar notificaciones asíncronas del estado de entrega de los SMS, como también iniciar y terminar notificaciones de acuse de recibo.
- Mensajería multimedia: Implementa la capacidad para enviar y recibir mensajes multimedia (MMS), proporcionando acceso de alto nivel a las funciones de manejo de MMS mediante protocolos provistos por la red.
- Pago: involucra las funcionalidades de pagos de reservas, pre-pagos y pos-pagos de servicios a los que el usuario esta suscrito.
- Administración de cuenta: gestiona las cuentas de los suscriptores, soportando consulta de saldo, recarga directa, y recarga a través de *vouchers*.
- Estado del terminal: permite a las aplicaciones obtener el estado de un terminal: alcanzable (dentro del área de cobertura), inalcanzable (fuera del área de cobertura) u ocupado. También permite que las aplicaciones reciban notificaciones cuando un terminal determinado cambia su estado.
- Ubicación del terminal: provee la funcionalidad para obtener información sobre la ubicación del un terminal determinado, y notificar cambios de ubicación, la cual queda determinada por una latitud, una longitud, la altitud y precisión.
- Gestión de llamada: Especifica cómo una llamada debe ser manejada para un número específico. Se incluyen acciones como: aceptar llamada de determinados números, bloquear llamadas de una lista de números, redireccionamiento de

llamadas hacia otro número para ciertas llamadas entrantes, iniciar un audio al recibir una llamada.

- Llamada de audio: Provee de entrega mensajes multimedia y de administración dinámica contenidos multimedia para los participantes involucrados en una llamada. Estos contenidos multimedia pueden ser: texto para ser usado en un motor Texto-a-voz, contenido en audios, VoiceXML ejecutado en un navegador que soporte esta funcionalidad, *video streaming*, y captura de contenidos multimedia.
- Conferencia multimedia: es un simple Servicio Web que permite la creación de conferencias multimedia y la administración dinámica de los participantes. Soporta múltiples flujos multimedia para establecer la comunicación entre los participantes. En particular, audio y video son comúnmente usados, incluyendo la dirección del flujo (entrada, salida, bidireccional).
- Administración de lista de direcciones: maneja grupos de suscriptores, gestionando de creación, eliminación, consultas y accesos sobre estos.
- Presencia: Este servicio permite que la información de presencia (del tipo “estoy en el computador”, “al teléfono”, etc.) pueda ser obtenida para un o mas usuarios y también que los usuarios registren sus estados de presencia. Una aplicación de mensajería instantánea es un típico ejemplo que hace uso de esta interfaz.

Actualmente se encuentra liberada la versión 3.0 de Parlay-X, disponible desde Junio del 2007 en los portales de ETSI, Parlay y 3GPP, la cual incluye, además de los servicios listados anteriormente, servicios como, envío masivo de mensajes, geocodificación, administración de calidad de servicio, configuración de dispositivos y sus capacidades, control de flujo (*streaming*) multimedia, y administración de sesiones multimedia multi-destino.

En el presente trabajo, se toma como referencia la versión 2.1 de Parlay-X, debido a que algunos de los servicios disponibles en esa versión son implementados en la última versión de un emulador desarrollado por la empresa de telecomunicaciones Ericsson, el que fue usado en el presente trabajo y se describe en el siguiente capítulo.

CAPÍTULO IV: ERICSSON SDK PARA SERVICIOS WEB DE TELECOMUNICACIONES.

4.1 Descripción del Kit de Desarrollo

Los Servicios Web en telecomunicaciones son usados para establecer comunicaciones del tipo *back-end*[16] y negocio-a-negocio (B2B), por ejemplo, entre una aplicación en el lado del servidor y una pasarela o *gateway* de Servicios Web en el entorno de un operador que expone capacidades de su red de telecomunicaciones, tales como SMS, MMS, ubicación de terminales, entre otras.

El SDK o kit de desarrollo de software de Servicios Web de telecomunicaciones puesto a disposición por Ericsson, está integrado por dos partes:

- Un conjunto de librerías de componentes JavaSE para Servicios Web especificados por Parlay-X.
- Un emulador de capacidades de redes de telecomunicaciones para poner a prueba las aplicaciones desarrolladas.

Cada una de estas partes será explicada en detalle en los siguientes puntos.

4.1.1 El emulador

El emulador de Servicios Web de telecomunicaciones está basado en Parlay-X versión 2.1 (ver punto 3.2), estándar que consiste en interfaces que definen, en un alto nivel, Servicios Web de telecomunicaciones. La versión del emulador descrita en el presente proyecto fue liberada en Abril del 2008 e implementada usando tecnología Java EE, por lo que para su ejecución se requiere del servidor de aplicaciones *Java Glassfish* o SJSAS (*Sun Java System Application Server*) versión 9.0. Alternativamente también el emulador puede correr sobre Tomcat 6.

El emulador implementa las APIs Parlay-X 2.1 y puede ser usado para probar aplicaciones Parlay-X sin necesidad de un *gateway* verdadero que esté funcionando en el entorno del operador.

[16] Comunicación Back-End: Comunicación establecida entre un servidor y los usuarios finales.

En la figura 17 se grafica la interacción del emulador con las aplicaciones que hacen uso de las capacidades de servicio emuladas, junto con las tecnologías involucradas:

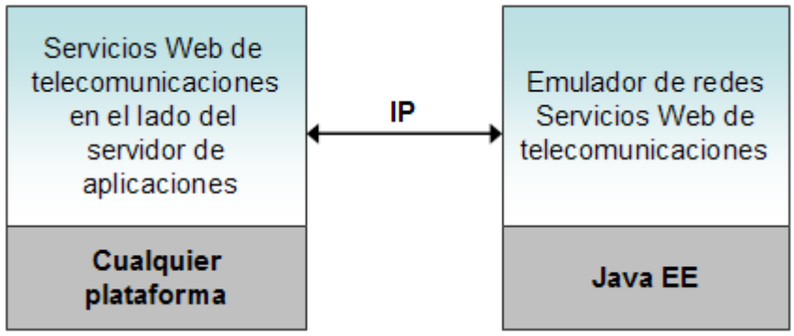


Figura 17. Emulador de Servicios Web para redes de telecomunicaciones interactuando con una aplicación.

El emulador implementa las siguientes capacidades de servicio basadas en las especificaciones de Servicios Web de Parlay-X 2.1. Además, se mencionan algunas aplicaciones en donde estas capacidades de servicio pueden ser usadas:

- SMS: por ejemplo, una aplicación que reciba SMS y mediante la cual los usuarios puedan publicar un artículo en un Blog.
- MMS: por ejemplo, para publicar contenidos multimedia en un Blog.
- WAP-Push: una aplicación podría enviar un link hacia de página WAP interesante a algún terminal móvil.
- *Terminal status*: por ejemplo, una aplicación podría comprobar si un terminal móvil esta prendido o no, antes de enviar un mensaje SMS.
- *Terminal location*: una aplicación que verifique la ubicación de un terminal móvil.

Como se puede observar, existen algunos Servicios Web especificados en Parlay-X 2.1 que no han sido incorporados. El emulador se basa en una implementación abierta, en donde el código fuente es incluido en el SDK, y posee un diseño flexible, dando la posibilidad de poder extenderlo con nuevas características.

4.1.2 Componentes Java SE

La gran ventaja que proporcionan los componentes Java SE incluidos en el SDK, es que sólo se requiere de tener conocimiento en tecnologías Java. De manera contraria, sin el uso de los componentes es necesario que los desarrolladores de aplicaciones tengan conocimiento sobre el estándar Parlay-X y sobre SOAP (*Simple Object Access Protocol*), protocolo que permite a las aplicaciones comunicarse con una pasarela o *gateway* de Servicios Web en el lado del operador, dando acceso a las capacidades de las redes de telecomunicaciones.

En la figura 18 se pueden observar las diferentes capas usadas por una aplicación para acceder a las funcionalidades de la red. También se señala la tecnología usada para comunicar una capa con otra.

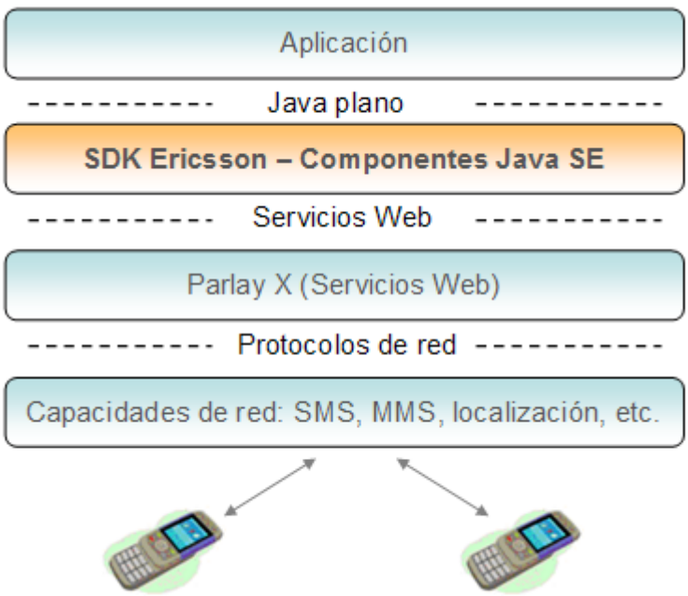


Figura 18. Capa del SDK de Ericsson para establecer comunicación entre una aplicación y los Servicios Web de Parlay-X .

Con el SDK para Servicios Web de telecomunicaciones y los componentes incluidos, es posible desarrollar aplicaciones sin preocuparse en los detalles de las interfaces de Parlay-X y las peticiones SOAP asociadas a *callbacks*, seguridad, archivos adjuntos a MMSs y otras características. Los componentes incluidos en el SDK, ayudan a desarrollar cualquier tipo de aplicación, ya sea una aplicación Java SE, aplicaciones Web, aplicaciones Enterprise *JavaBeans* y aplicaciones distribuidas en clusters.

Los componentes Java SE están basados en tecnología *JavaBeans* y fueron diseñados para entregar a los desarrolladores de aplicaciones, facilidad de uso y flexibilidad. Poseen la ventaja de integrarse en una arquitectura MVC (Modelo Vista Controlador) en contenedores Web de Java EE, además de entregar soporte para aplicaciones distribuidas y en clusters.

Las siguientes tecnologías fueron usadas para la implementación de los componentes Java SE:

- API Java para Servicios Web XML (JAX-WS) en Java SE 6.0 update2.
- JavaMail 1.4.1.
- XML y Seguridad en Servicios Web (XWSS) disponibles en el pack JWSDP 2.0 (Pack para desarrolladores de Servicios Web Java)
- SAAJ (SOAP *Attachments* API para Java) disponible en el pack JWSDP 2.0.

El conjunto de componentes incluidos en el SDK está compuesto por los siguientes elementos:

- Componentes *JavaBeans*: componentes reusables, flexibles y fáciles de usar, utilizados para tener acceso a los Servicios Web disponibles implementados en el emulador.
- Componentes *Callback*: posibilitan el manejo de *callbacks* o llamadas de vuelta desde el emulador hacia las aplicaciones.

Al igual que el emulador, estos componentes fueron creados bajo una implementación abierta, en donde el código fuente es incluido.

4.2 Software requerido por el emulador

El emulador de Parlay-X se encuentra disponible para ser descargado libremente en la página oficial de Ericsson, previo registro del usuario en esta página. Para ponerlo en marcha, es necesario instalar terceros componentes, los que se listan a continuación:

- Java EE 5 con JDK update 2 (incluye *Java Glassfish*)
- Apache Ant.
- Apache Tomcat 6.0 (opcional: sólo si el emulador es desplegado con Tomcat. Si así, se requiere también de JSTL, *Java Standard Tag Library*).

Los pasos realizados para configurar e instalar el emulador se encuentran debidamente explicados en el manual del emulador disponible en el sitio oficial de Ericsson, sección "*Telecom Web Services*"[17].

4.3 Interfaz de usuario del emulador

Una vez que el emulador haya sido configurado e instalado, se procede con la ejecución de éste sobre el servidor de aplicaciones *Java Glassfish*, mediante el uso de un navegador como, por ejemplo, *Mozilla Firefox* o *Internet Explorer*.

4.3.1 Interfaz gráfica principal

La interfaz principal del emulador se muestra a continuación:

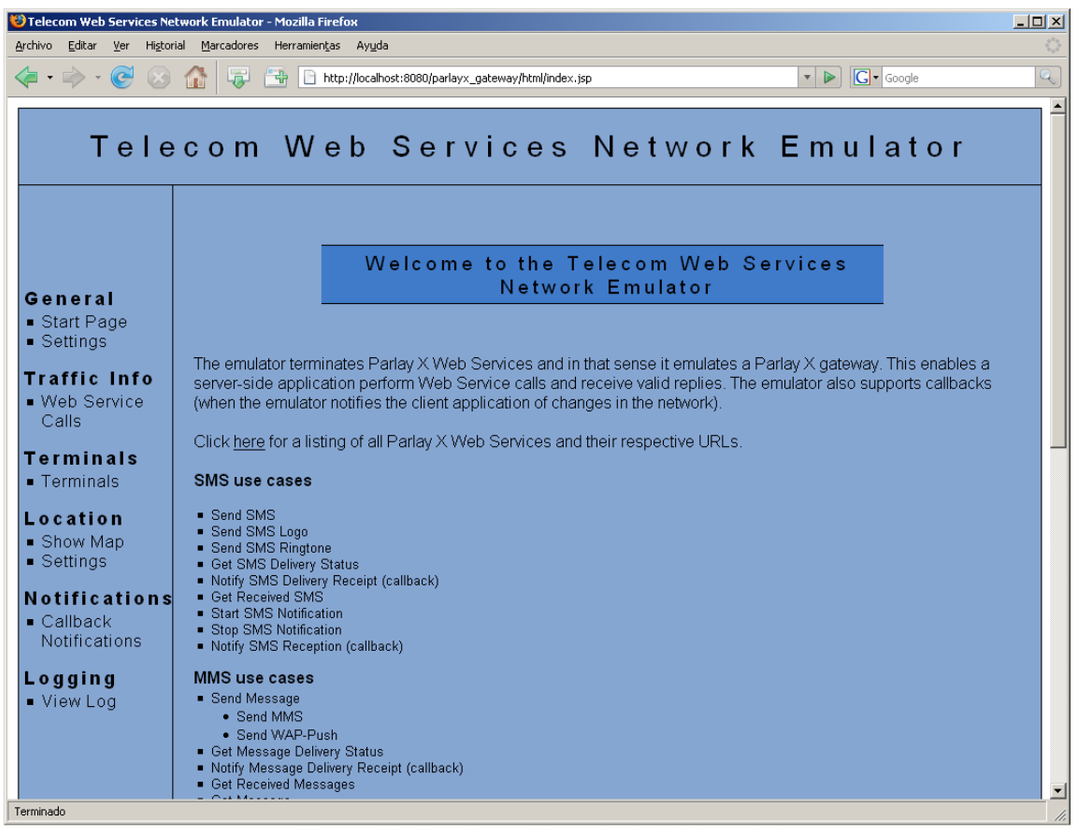


Figura 19. Interfaz gráfica principal del emulador.

[17] http://www.ericsson.com/mobilityworld/sub/open/technologies/parlayx/index.html

En esta interfaz, se describen los casos de uso en los cuales el emulador puede utilizarse, además de una opción para listar, junto con las URLs respectivas, todos los Servicios Web de Parlay-X implementados los que son mostrados a continuación:

Tabla 3. Servicios Web de Parlay-X implementados en el emulador.

Servicio Web	**Información**	**URL**
SendSms	Address:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/SendSms
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/SendSms?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.sms.SendSmsWsEndpoint
SmsNotificationManager	Address:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/SmsNotificationManager
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/SmsNotificationManager?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.sms.SmsNotificationManagerWsEndpoint
ReceiveSms	Address:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/ReceiveSms
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXSmsAccess/services/ReceiveSms?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.sms.ReceiveSmsWsEndpoint
SendMessage	Address:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/SendMessage
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/SendMessage?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.mms.SendMessageWsEndpoint
MessageNotificationManager	Address:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/MessageNotificationManager
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/MessageNotificationManager?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.mms.MessageNotificationManagerWsEndpoint
ReceiveMessage	Address:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/ReceiveMessage
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXMmmAccess/services/ReceiveMessage?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.mms.ReceiveMessageWsEndpoint
TerminalStatusNotificationManager	Address:	http://localhost:8080/parlayx_gateway/ParlayXTsAccess/services/TerminalStatusNotificationManager
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXTsAccess/services/TerminalStatusNotificationManager?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.ts.TerminalStatusNotificationManagerWsEndpoint

Servicio Web	Información	URL
TerminalStatus	Address:	http://localhost:8080/parlayx_gateway/ParlayXTsAccess/services/TerminalStatus
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXTsAccess/services/TerminalStatus?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.ts.TerminalStatusWsEndpoint
TerminalLocation	Address:	http://localhost:8080/parlayx_gateway/ParlayXTlAccess/services/TerminalLocation
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXTlAccess/services/TerminalLocation?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.tl.TerminalLocationWsEndpoint
TerminalLocationNotificationManager	Address:	http://localhost:8080/parlayx_gateway/ParlayXTlAccess/services/TerminalLocationNotificationManager
	WSDL:	http://localhost:8080/parlayx_gateway/ParlayXTlAccess/services/TerminalLocationNotificationManager?wsdl
	Implementation class:	parlayxws.emulator.ws.px_spec_v2_1.tl.TerminalLocationNotificationManagerWsEndpoint

4.3.2 Sección de configuración

Antes de usar las funcionalidades del emulador, es necesario realizar algunas configuraciones en la sección "*Settings*" del menú "General". Esta configuración consiste en ajustar el modo de compatibilidad de Parlay-X a "Parlay X 2.1" (la otra opción es "SIG 4.0", *gateway* de Parlay/OSA desarrollada también por Ericsson).

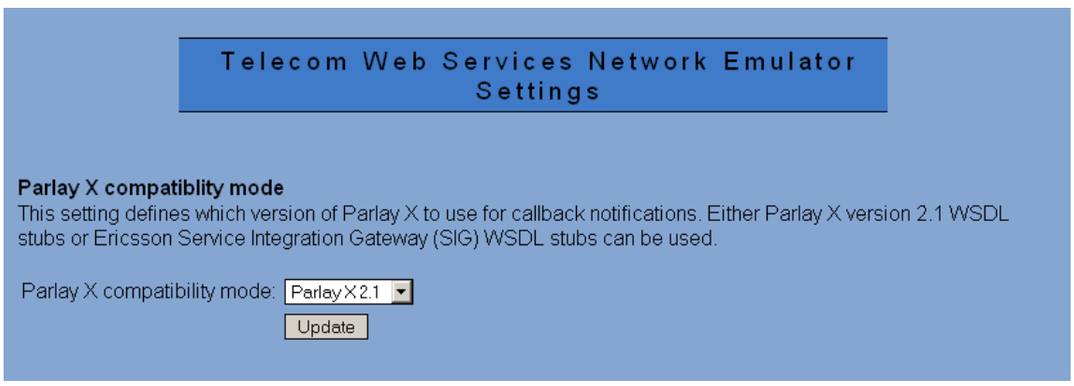


Figura 20. Modo de compatibilidad de Parlay-X.

En la sección "*Settings*" también es posible emular excepciones Parlay-X, las que son lanzadas cuando una aplicación envía un requerimiento al emulador que excede su nivel de capacidad de servicio (SLA), o se solicita un Servicio Web que no está disponible.

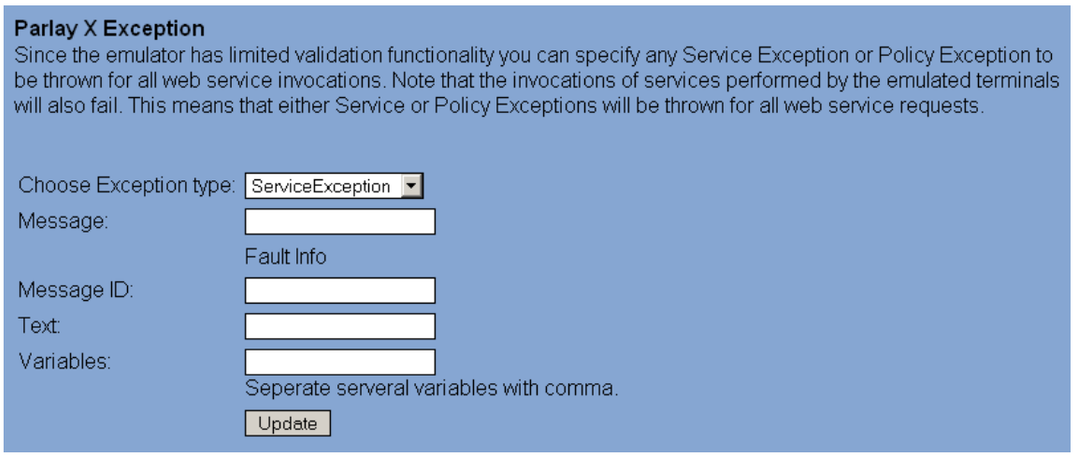


Figura 21. Emulación de una excepción de Parlay-X

Existen dos excepciones SOAP especificadas en Parlay-X: "*Service Exception*" y "*Policy Exception*". En la figura anterior, se muestra el formulario de ingreso de parámetros para ingresar una excepción. Esta característica del emulador puede ser usada para probar el comportamiento de una aplicación cuando ésta recibe una excepción en respuesta a la llamada de un servicio Web de Parlay-X.

4.3.3 Terminales móviles de prueba

En la sección "*Terminals*" es posible agregar y remover terminales móviles emulados para realizar pruebas de envío de mensajes SMS, MMS, WAP-Push, ubicación y estado de terminales.

La información para cada terminal es mostrada en una tabla, en donde se indica información como el número que identifica a cada terminal, la cantidad de mensajes recibidos, el estado (alcanzable, ocupado, fuera del área de cobertura), la opción para mostrar en una ventana del navegador la interfaz del terminal y terminar su emulación. Toda esta información se presenta de la siguiente forma:

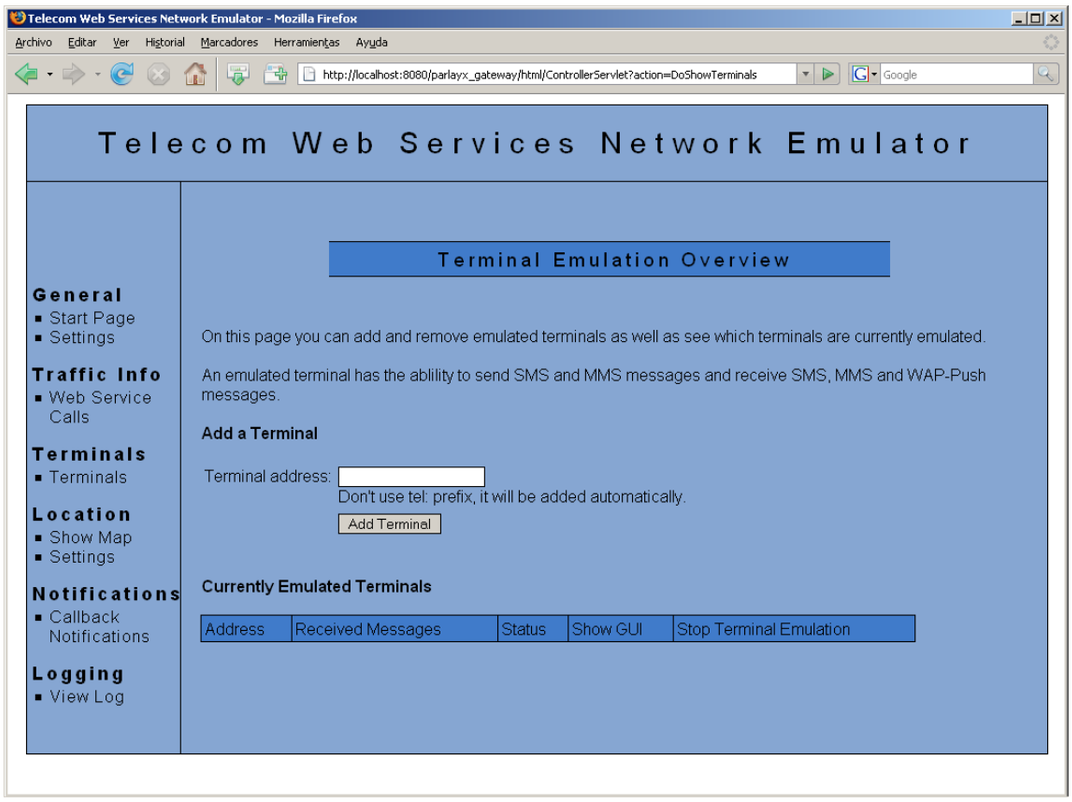


Figura 22. Interfaz de emulación de terminales.

Para agregar nuevos terminales móviles, se debe ingresar el número en el campo "*Terminal address*" y luego presionar en "*Add Terminal*". A modo de ejemplo se agregaron dos terminales con números "111" y "222" Los terminales agregados se listan de la siguiente forma:

Currently Emulated Terminals

Address	Received Messages	Status	Show GUI	Stop Terminal Emulation
tel:111	0	Reachable	show GUI	stop terminal emulation
tel:222	0	Reachable	show GUI	stop terminal emulation

Figura 23. Lista de terminales emulados.

Para visualizar las interfaces de los terminales emulados se debe presionar en la opción "*show GUI*". Estos terminales se ejecutarán en ventanas tipo *pop-up* del navegador y tienen la siguiente apariencia:

Figura 24. Interfaces gráficas de los terminales emulados.

En cada terminal es posible establecer el estado de estos, ya sea "Apagado o fuera del área de cobertura", presionando en "*Off*", o el estado de "Ocupado" presionando en "*Make a call*".

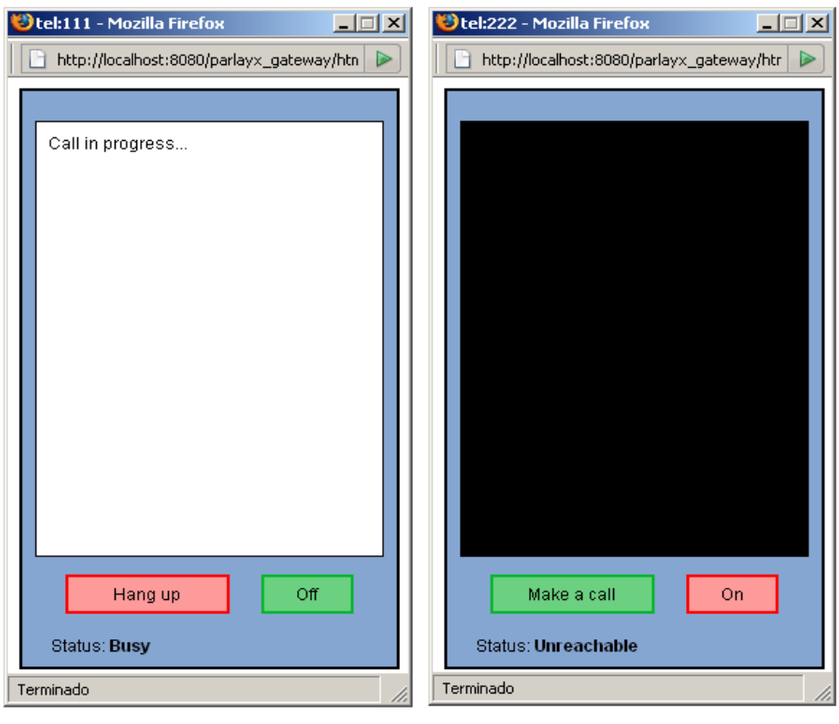


Figura 25. Estados posibles de los terminales móviles.

En el menú principal de cada terminal (figura 24) existen las opciones "*Messaging*" y "*Multimedia*", la primera para visualizar la bandeja de entrada de mensajes SMS y MMS, y enviar estos tipos de mensajes; y la segunda para administrar el contenido multimedia en el terminal, usado al momento de enviar/recibir mensajes MMS.

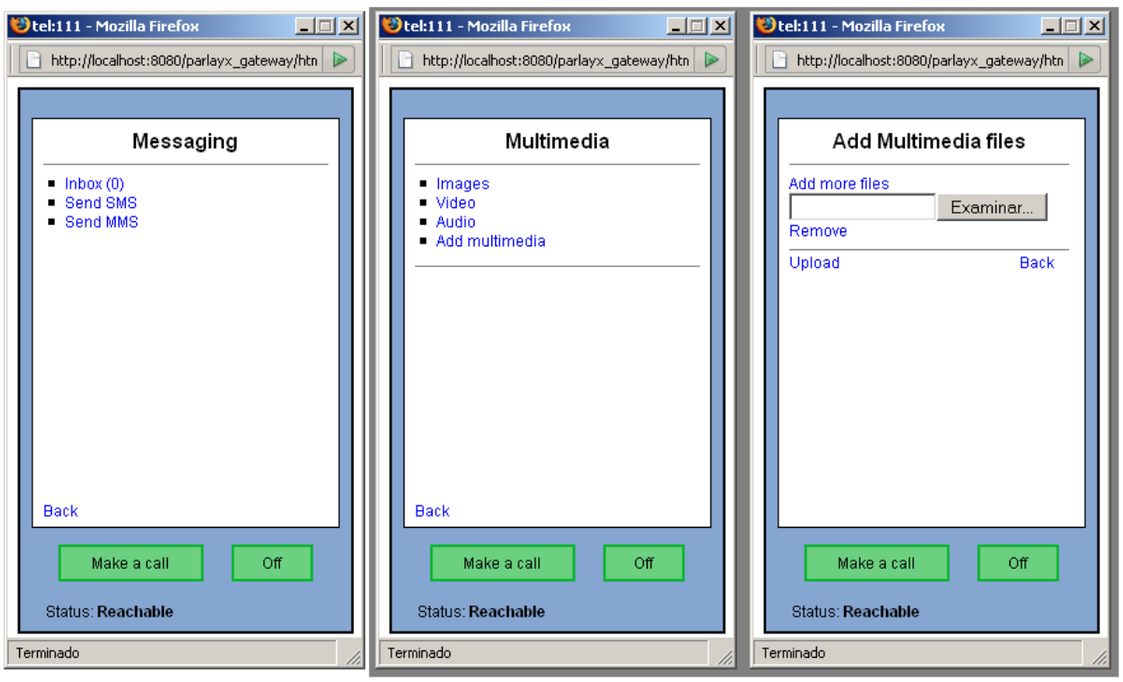


Figura 26. Opciones de "Messaging" y "Multimedia", y agregar contenido multimedia al terminal.

De esta forma, las aplicaciones creadas que implementen las funcionalidades SMS y/o MMS pueden hacer uso de los terminales emulados para poner a prueba la aplicación.

A modo de ejemplo, se envía un mensaje SMS desde un terminal con número "111" hacia el terminal "222", ambos creados en el emulador. Para esto, en el terminal "111", se selecciona la opción "*Messaging*" del menú principal, y luego se presiona "*Send SMS*". Se completa el campo "*To*", escribiendo el número del destinatario, en este caso, "222", se escribe el mensaje y se envía presionando "*Send*".

La acción de enviar y recibir el mensaje queda graficada en la figura 27.

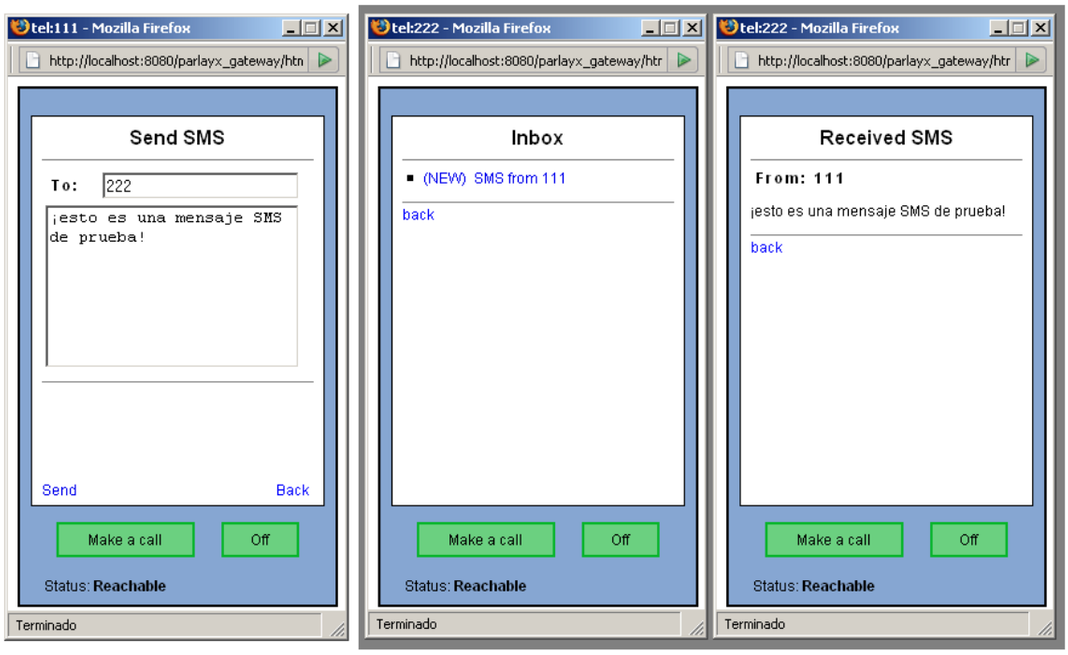


Figura 27. Envío de un mensaje SMS desde un terminal a otro.

4.3.4 Servicio de localización de terminales

En la sección "*Location*", opción "*Settings*", se realiza la configuración del servicio de localización del emulador, el cual mediante un mapa se emula la ubicación de los terminales creados. La configuración consiste en asignar parámetros de borde para el mapa en uso, siendo éstos la longitud izquierda, longitud derecha, latitud superior y latitud inferior.

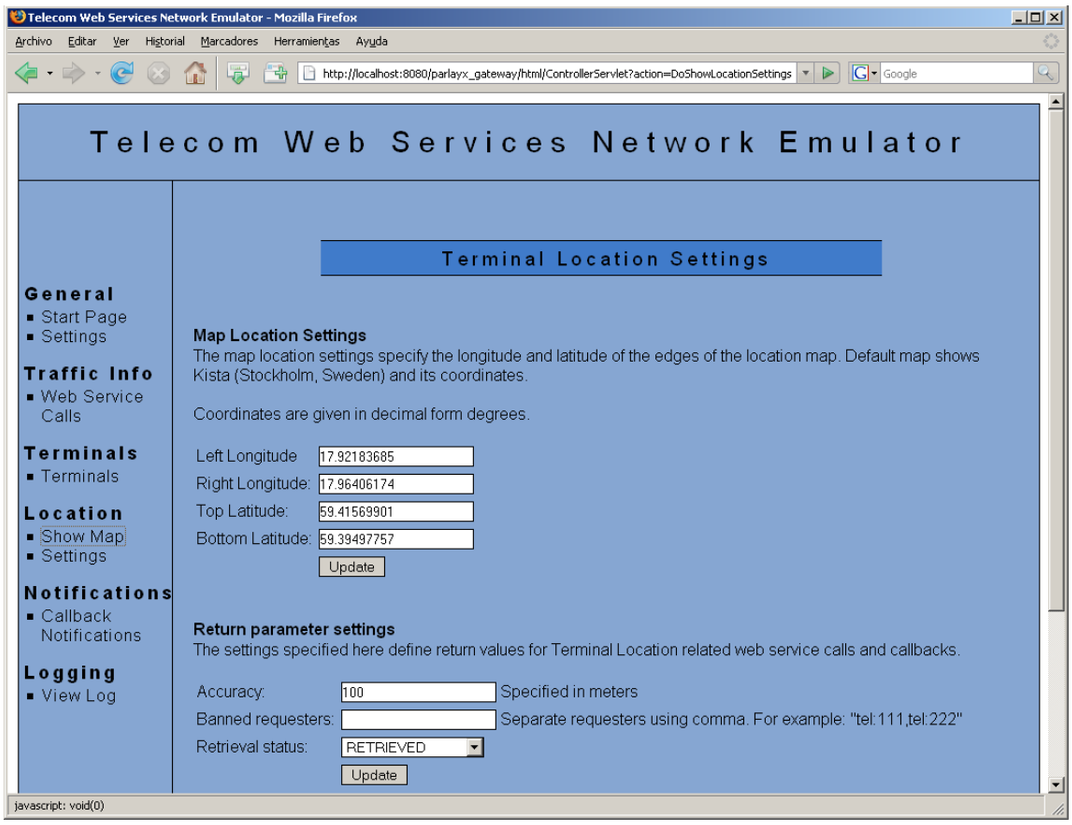


Figura 28. Configuración de los parámetros del mapa utilizado en la emulación de la localización de terminales.

Para visualizar el mapa, se selecciona la opción "*Show Map*" en la sección "*Location*". El mapa se muestra en la figura 29:

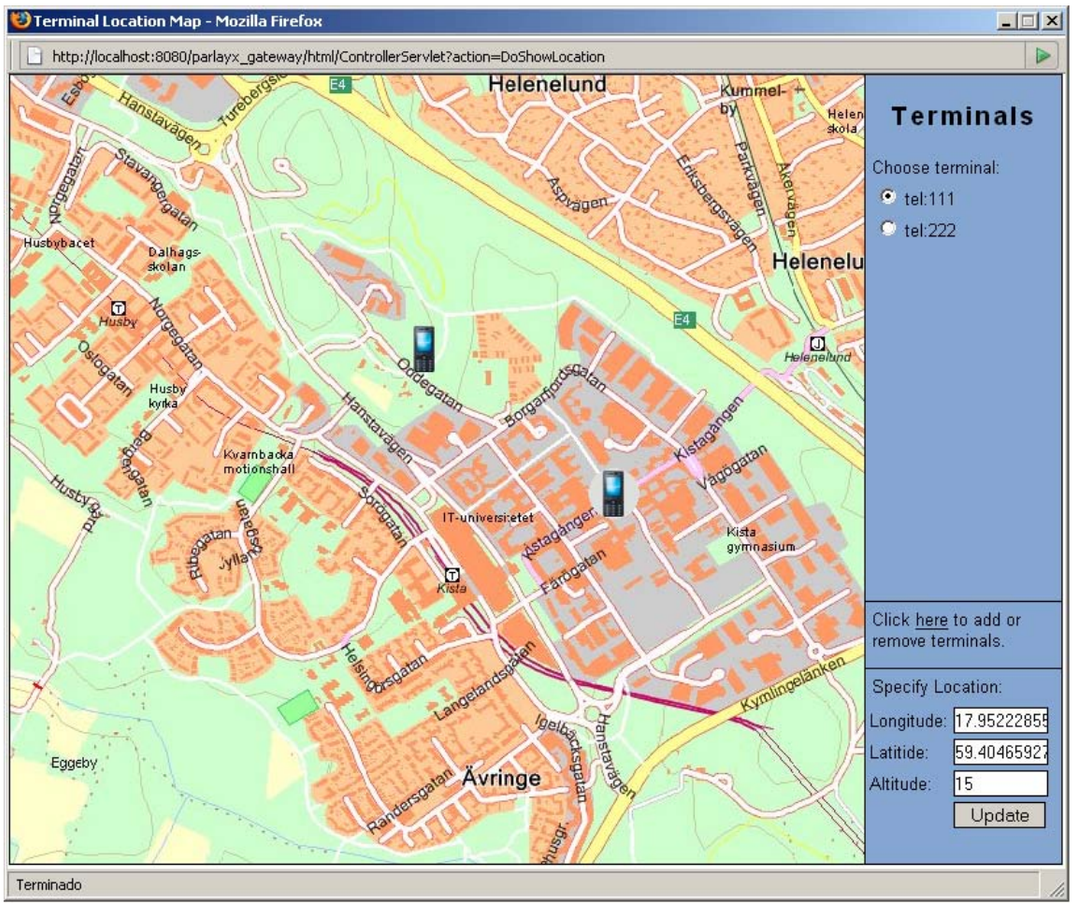


Figura 29. Mapa utilizado en la emulación de localización de terminales móviles

Aquí, es posible asignar manualmente los parámetros de ubicación de cada terminal, o también es posible, usando el *mouse*, arrastrar cada terminal hacia una posición determinada, actualizándose en forma dinámica la información de localización mostrada al costado derecho.

Con la emulación de localización de terminales se pueden crear y poner a prueba interesantes e innovadoras aplicaciones que hagan uso de este tipo de información. El emulador, entrega los Servicios Web apropiados para obtener e ingresar información de localización de un terminal determinado desde las aplicaciones que hacen uso de estos servicios.

CAPÍTULO V: PAM, UN SERVICIO PARLAY/OSA.

5.1 PAM: Presence and Availability Management.

5.1.1 Introducción

Es común tener situaciones en que un usuario trata de contactar sin éxito a otro, sin saber dónde éste se encuentra, ni en qué momento atenderá las llamadas o contestará los mensajes SMS. La experiencia de comunicación podría ser gratamente mejorada si un usuario pudiera hacer saber a sus contactos si éste se encuentra en línea y disponible, y el medio de comunicación por el que éstos prefieren que sean contactados. Esta información de disponibilidad y presencia de los usuarios se extiende sobre múltiples servicios y protocolos dedicados, los que actualmente son usados directamente por las aplicaciones como soluciones particulares para manejar este tipo de información. PAM o "Manejo de Presencia y Disponibilidad", pretende establecer un estándar para el mantenimiento, recuperación y publicación de información de presencia y disponibilidad a través de múltiples servicios y redes, para que, finalmente, esta información pueda ser manejada por las aplicaciones de una forma estandarizada. Los beneficios para el usuario final claramente están envueltos en una mejora en la experiencia de comunicación, ya que PAM permite que los usuarios contacten de manera más precisa y previsible a sus contactos.

El documento de especificación de la API de "*Precense and Availability Management*" [Ets05b] fue armonizado con el "modelo para presencia" [Day00] (RFC 2778) establecido por la IETF, y guarda consistencia con el trabajo realizado por la 3GPP para definir requerimientos y una arquitectura para un servicio de presencia estándar en la red.

El principal objetivo de PAM es facilitar el desarrollo de aplicaciones y servicios que se expandan sobre múltiples sistemas de comunicación (mensajería instantánea, email, fax, telefonía, etc.) y proveer al usuario final gran flexibilidad y control en el manejo de sus comunicaciones. Una plataforma estandarizada permite a los desarrolladores de software crear aplicaciones de administración de comunicación que son independientes de las tecnologías y protocolos subyacentes [Ets05b].

PAM, servicio a implementar en el presente trabajo, está especificado en las APIs de Parlay/OSA versión 5.0.

5.1.2 Conceptos claves

Existen algunos conceptos claves involucrados en las especificaciones de PAM. Estos se describen a continuación:

- **Identidad**: es una representación electrónica limitada que de una entidad (por ejemplo, una persona o una organización) que participa en las aplicaciones y servicios que implementan a PAM. La principal característica de una identidad es su nombre (o *handle*) por el cual las entidades son identificadas en las aplicaciones y servicios, y está compuesto de un dominio y un identificador, como por ejemplo, el nombre de una identidad podría ser "casa:usuario1" (dominio o *namespace* : identificador del usuario u organización).

- **Agente**: es una representación electrónica limitada de un dispositivo hardware (PDA, teléfono móvil, *notebook*, etc.), a través del cual las identidades se manifiestan o muestran su disponibilidad a las aplicaciones y servicios. Una característica importante de un agente es una lista de una o más capacidades asociadas a él, siendo la capacidad lo que hace usable a un agente. Ejemplos de capacidades son: mensajería instantánea, SMS, WAP, voz, entre otras. También, una capacidad puede representar la habilidad de un agente para reportar información relevante como por ejemplo, ubicación, velocidad, temperatura, etc.

- **Presencia**: este concepto ha sido usado en diversas áreas de aplicación, siendo mayormente explícita en aplicaciones de mensajería instantánea. Comenzando con una simple noción de estado *"online/offline"*, ésta se ha expandido a otro tipo de información, en un contexto de "estado de disposición" tal como "almorzando", "lejos del computador", etc., y en un contexto de "estado de actividad", como por ejemplo, "ocupado" o "sin nada que hacer (*idle*)". La información dentro de un contexto de "ubicación" o "localización" también es considerada dentro del concepto de presencia especificado en PAM.

- **Disponibilidad**: es una propiedad de una identidad que denota la habilidad y voluntad para compartir información sobre ella misma, o para comunicarse con otras identidades basándose en factores tales como el modo de comunicación requerida, como por ejemplo: "redireccionar al buzón de mensajes por voz si estoy fuera de cobertura", o qué personas pueden contactar a un usuario: "fuera de mis colegas, sólo mi jefe puede contactarme los fin de semanas", o, en mensajería instantánea, "estoy disponible sólo para determinados usuarios". El

concepto de presencia es un prerrequisito para el concepto de disponibilidad, así, un usuario podría estar presente pero no disponible.

Es importante señalar que mientras las especificaciones de PAM brindan un mecanismo para asociar preferencias con una identidad con la finalidad de que controlen su disponibilidad, éstas no especifican ni la sintaxis y semántica de las preferencias, ni el proceso por el cual la disponibilidad es procesada. Esos aspectos son dejados para la implementación.

- **Eventos**: son representaciones de determinadas ocurrencias relacionadas con los conceptos descritos anteriormente. PAM provee del registro de *callbacks* (llamadas recíprocas desde la pasarela Parlay/OSA hacia las aplicaciones) para notificar algunas ocurrencias. Ejemplos de eventos podrían ser:
  - Creación/eliminación de una identidad
  - Asociación de una instancia de agente con una identidad.
  - Cambio del estado de presencia o ubicación de una instancia agente.
  - Cambio en la información de presencia de una identidad
  - Cambio en la disponibilidad de un agente para una determinada forma de comunicación.

5.1.3 Interfaces de clases y diagramas UML

5.1.3.1 Interfaces de clases que componen a PAM

En el punto 3.1.1 se describió la arquitectura conceptual de Parlay/OSA definiendo los elementos que la componen, entre ellos su principal elemento: la pasarela o *gateway*, el cuál está compuesto por un *framework* y por servidores de capacidades de servicio (SCSs) que integran, finalmente, a las capacidades de servicios (SCFs). Según el documento correspondiente a la "parte 14" de las especificaciones de Parlay/OSA versión 5.0 [Ets05b], PAM consiste de las siguiente SCFs, compuestas por un conjunto de interfaces:

- PAM *Provisioning Service*: consiste en interfaces para manejar agentes, identidades y las relaciones entre estos.

- PAM *Access Service*: consiste en interfaces para obtener y actualizar información de presencia y disponibilidad.

- PAM *EventManagement Service*: consiste en interfaces para suscribir eventos en PAM y para notificar tales eventos.

5.1.3.2 Esquema de colores y nombres para las clases de interfaces

Las interfaces de PAM son modeladas a través de diagramas UML de clases. La construcción de estos diagramas utiliza el siguiente esquema de colores [Ets05a]:

- Azul: para las interfaces de aplicación (cliente).
- Amarillo: para las interfaces de las SCFs de PAM.

Además, las interfaces siguen un esquema de nombres basado en las siguientes reglas:

- Interfaces con prefijo "`IpApp`": son interfaces de aplicación utilizadas para poder realizar las llamadas recíprocas o *callbacks* desde PAM hacia las aplicaciones. Este tipo de interfaces necesitan ser implementadas por una aplicación cliente.
- Interfaces con prefijo "`IpClient`": son interfaces que se deben implementar en las aplicaciones cliente y que son compartidas junto con las interfaces de una SCF.
- Interfaces con prefijo "`IpFw`": son interfaces implementadas en el *framework* y son usadas por una SCF.
- El resto de interfaces con prefijo "`Ip`" descritas en el presente trabajo pertenecen a las capacidades de servicio que componen a PAM.

5.1.3.3 SCF PAM Provisioning: Interfaces de clases y diagrama UML

La capacidad de servicio PAM *Provisioning* contiene las siguientes interfaces de servicio:

- `IpPAMIdentityManagement`: el propósito de esta interfaz es gestionar los nombres de los suscriptores, aliases, grupos, y un conjunto de atributos llamados *Profiles* asociados a cada suscriptor (identidad).
- `IpPAMAgentManagement`: su objetivo es administrar nombres de agentes, comunicación y capacidades, y un conjunto de atributos llamados *Profiles* asociados con agentes.
- `IpPAMAgentAssignment`: su propósito es administrar la relación entre identidades y los agentes asignados a éstas.
- `IpPAMIdentityTypeManagement`, `IpPAMAgentTypeManagement` y `IpPAMCapabilityManagement`: Estas tres interfaces permiten la definición de tipos de identidades, tipos de agentes y capacidades de los agentes usadas para el manejo de presencia y disponibilidad.

Las referencias de las interfaces listadas anteriormente son administradas y obtenibles a través de la interfaz `IpPAMProvisioningManager`, la cual es el único punto de entrada a la SCF *Provisioning* para las aplicaciones y la única interfaz de PAM *Provisioning* que puede ser descubierta por el *framework*.

Las interfaces descritas recientemente y sus relaciones pueden ser representadas a través del siguiente diagrama de clases UML:

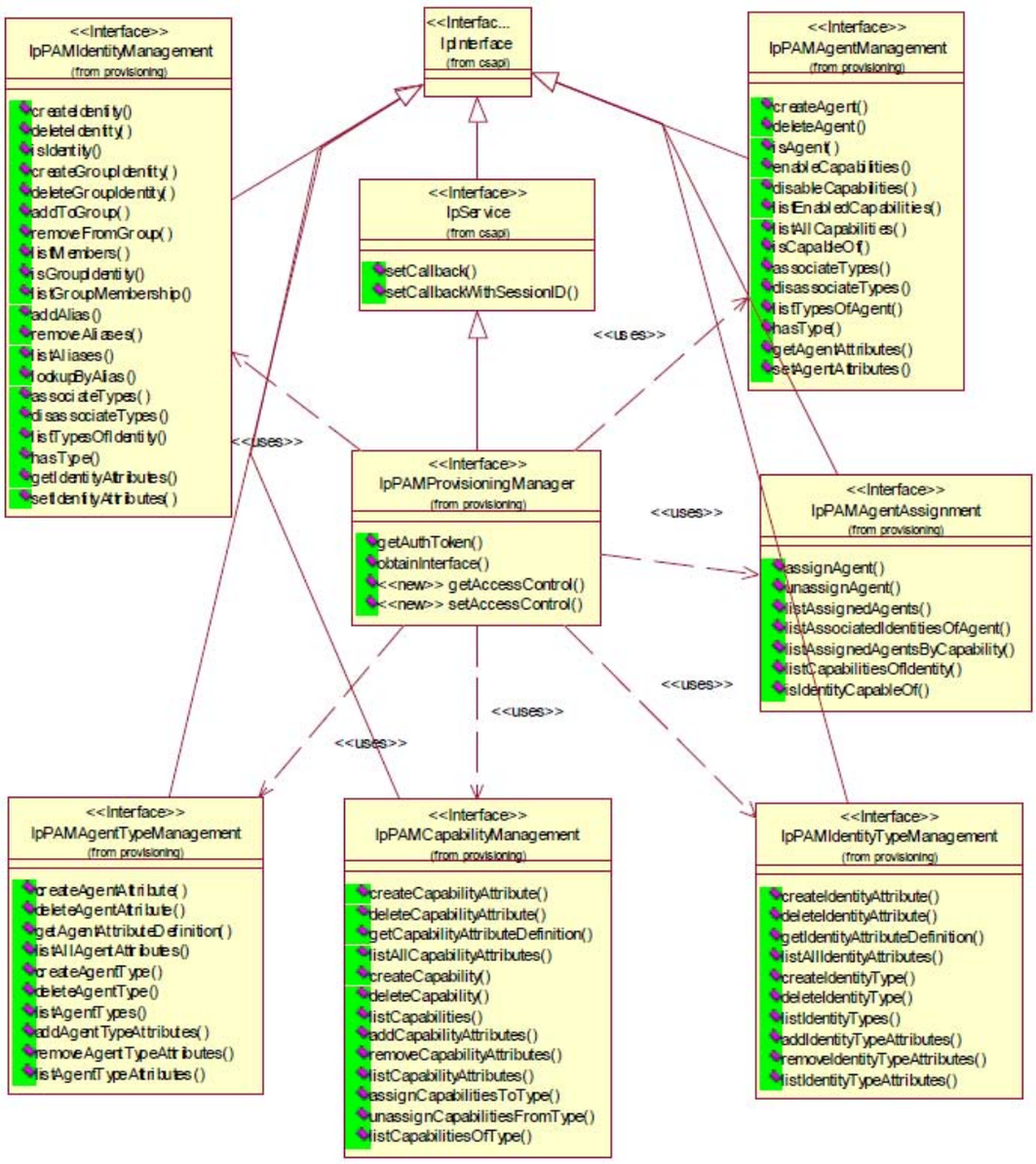


Figura 30. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Provisioning.

5.1.3.4 SCF PAM Access: Interfaces de clases y diagrama UML

La capacidad de servicio PAM *Access* está compuesta por dos paquetes: uno contiene las interfaces de aplicación, y el otro las interfaces de servicios. El paquete de interfaces de aplicación de PAM *Access* consiste en cero o más instancias de la interfaz `IpAppPAMPreferenceCheck` implementada por las aplicaciones clientes para que PAM pueda controlar acceso y procesar disponibilidad en la aplicación. El paquete de interfaces de servicio consiste en simples instancias de las siguientes interfaces:

- `IpPAMAgentPresence`: Con esta interfaz es posible mantener información de agentes de manera dinámica.
- `IpPAMIdentityPresence`: El propósito de esta interfaz es mantener información dinámica de identidades.
- `IpPAMAvailabilityManagement`: Está interfaz tiene dos funcionalidades: i) Administrar las preferencias especificadas para la disponibilidad de una identidad, y ii) realizar consultas de disponibilidad de identidades para determinadas capacidades (SMS, voz, mensajería instantánea, etc.)

Las referencias a las interfaces listadas anteriormente son administradas y obtenibles a través de la interfaz `IpPAMPresenceAvailabilityManager`, la cual es el único punto de entrada a la SCF *Access* para las aplicaciones y la única interfaz de PAM *Access* que puede ser descubierta por el *framework*.

Las interfaces descritas anteriormente y sus relaciones quedan representadas a través del siguiente diagrama de clases UML:

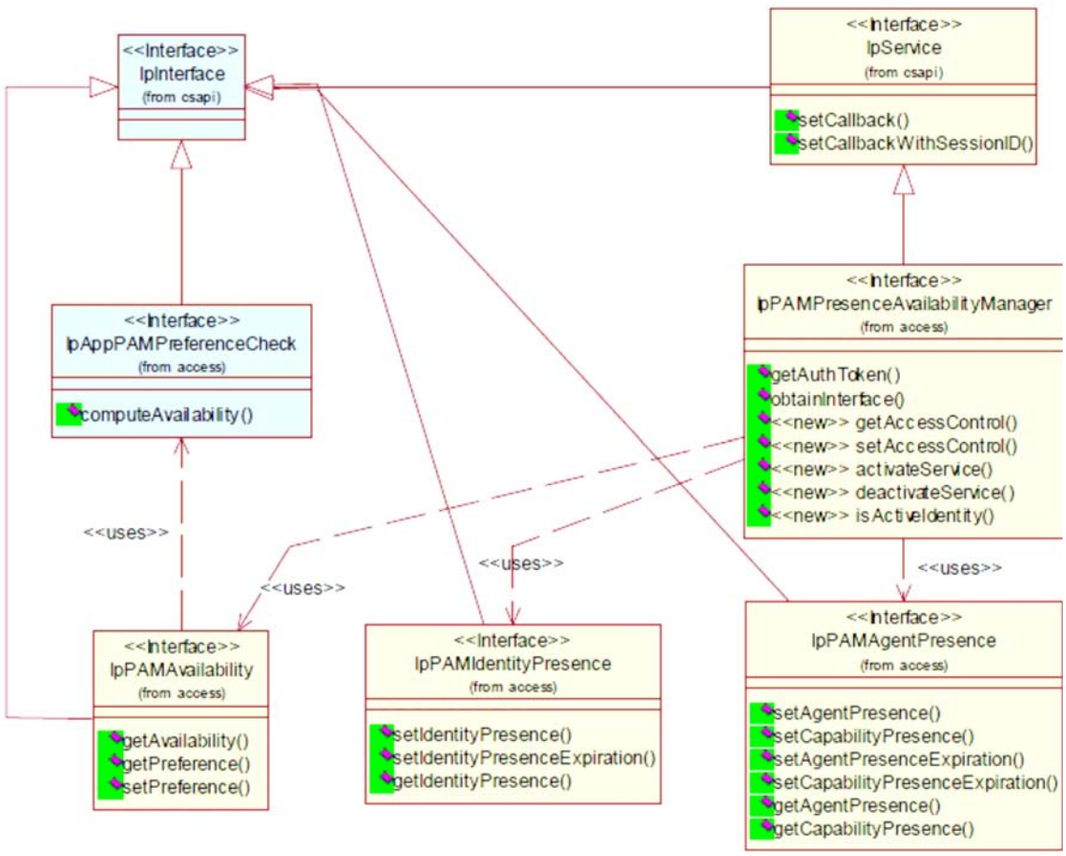


Figura 31. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Access.

5.1.3.5 SCF PAM Event: Interfaces de clases y diagrama UML

La capacidad de servicio o SCF de PAM *Event Management* consiste en dos paquetes, uno para las interfaces de aplicación y otro para las interfaces de servicio. El paquete de aplicación de PAM *Event Management* consiste en cero o más instancias de la interfaz `IpAppPAMEventHandler` la cual debe ser implementada por una aplicación para registrarse en PAM y poder recibir notificaciones de eventos. El paquete de servicios consiste en una simple instancia de la interfaz `IpPAMEventHandler`, obtenida por la aplicación a través de la interfaz de servicio de `IpPAMEventManager`, correspondiente al único punto de entrada a PAM *Event*, y a la única interfaz que puede ser descubierta por el *framewok*.

- `IpPAMEventHandler`: el propósito de esta interfaz es administrar los registros de eventos y de interfaces de las aplicaciones clientes para notificaciones posteriores. Todas las notificaciones son enviadas de manera asíncrona después de la ocurrencia de un evento.

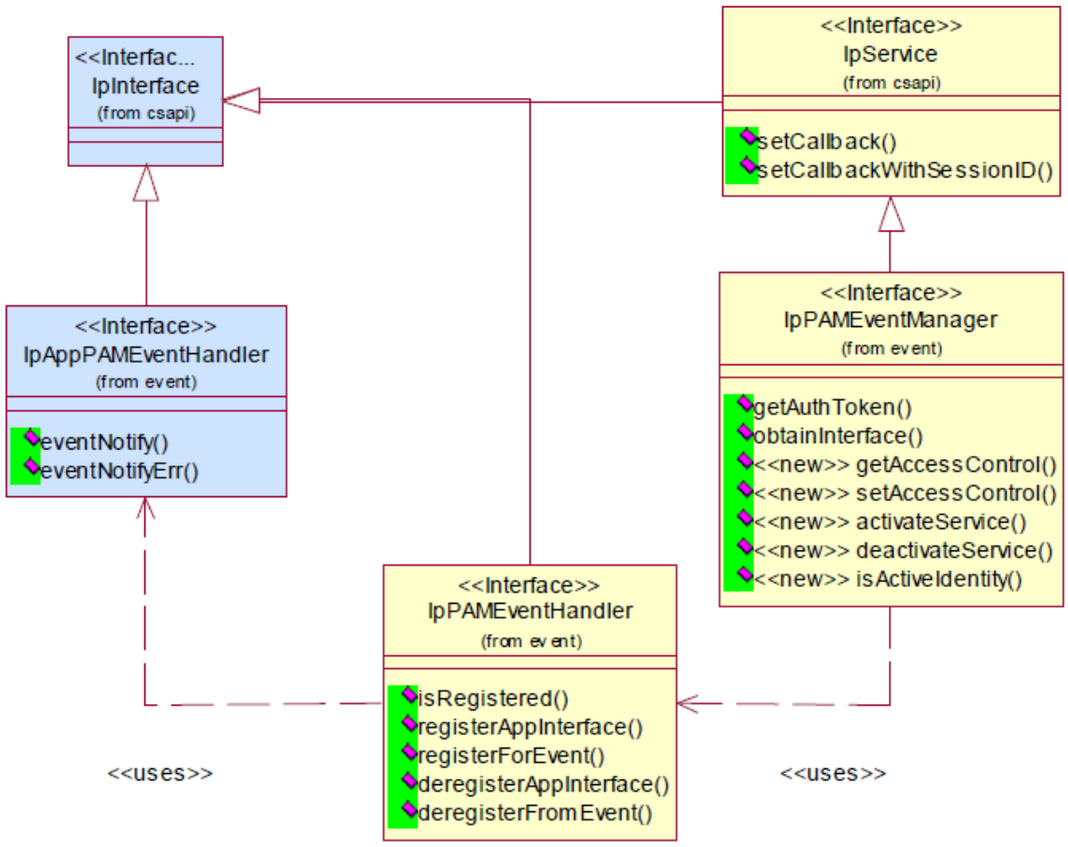


Figura 32. Diagrama de clases UML para las interfaces que componen la capacidad de servicio PAM Event.

Las interfaces que integran a cada SCF de PAM, son finalmente utilizadas de manera remota por las aplicaciones Parlay/OSA que se desarrollen, para comunicarse con la pasarela o *gateway*, y poder hacer uso del servicio de "Manejo de Presencia y Disponibilidad" implementado.

5.2 Un Framework para Parlay/OSA

5.2.1 Descripción

Recordando de manera breve lo descrito en los puntos 3.1.1.3 y 3.1.1.4, el *framework* corresponde a una parte fundamental en la arquitectura de Parlay/OSA y cumple una función de intermediario entre las aplicaciones clientes y las SCFs implementadas en el *gateway* Parlay/OSA, realizando labores de autenticación de las aplicaciones y de descubrimiento y registro de los SCFs en la pasarela, para que estas capacidades de servicios puedan ser descubiertas y utilizadas por una aplicación.

En el presente trabajo, se utilizó un *framework* desarrollado por tres estudiantes de la universidad RWTH-Aachen, Alemania, en el año 2005, cuya implementación se basa en el documento de especificación "parte 3" de las APIs de Parlay/OSA versión 5.0 [Ets05c].

5.2.2 Tecnologías empleadas

Las tecnologías utilizadas en la implementación del *framework* se listan a continuación:

- JacORB versión 2.2: implementación del estándar CORBA y compilador IDL-to-Java requerido para la generación de archivos Java (interfaces, clases *Holders* y *Helpers*, *Stubs* y *Skeletons*) a partir de los archivos IDL adjuntos en la especificación de Parlay/OSA, "parte 3": "*framework*" [Ets05c].
- *Java 2 Standard Edition* versión 1.5: utilizado en la implementación de las interfaces Java del *framework* Parlay/OSA generadas por el compilador JacORB.
- MySQL versión 5.0: base de datos usada para registrar logs de actividad y capacidades de servicio o SCFs disponibles en el gateway de Parlay/OSA.

5.2.3 Configuraciones para la puesta en marcha del framework

El código fuente del *framework* fue obtenido realizando un *checkout* desde un repositorio CVS instalado en un servidor Sun SPARC con SunOS, ubicado en las dependencias de la universidad alemana. El *checkout* fue efectuado utilizando el entorno de desarrollo Eclipse versión 3.1.1, con los datos de conexión entregados oportunamente.

El código fuente fue descargado, mediante el *checkout*, a una máquina de desarrollo local, en donde posteriormente se llevo a cabo la puesta en marcha del *framework* y la implementación y prueba final de las tres SCFs que componen a PAM.

El proyecto "*framework*" quedó compuesto por la siguiente estructura de directorios:

- `build/`: archivos `.class` compilados a partir del código fuente del *framework*.

- `dist/`: se aloja el código Java distribuible, consistente en archivos `jar` que contienen las clases Java compiladas del proyecto.
- `docs/`: documentación referente a la especificación del *framework* de Parlay/OSA 5.0 y Javadocs correspondiente.
- `etc/`: archivos de configuración utilizados en la conexión del *framework* con la base de datos.
- `idl/`: contiene los archivos IDL de CORBA que definen las interfaces especificadas para el *framework*.
- `sql/`: contiene *scripts* para crear una tabla de logs y otra para registrar los SCFs soportados por el *framework*.
- `src/`: código fuente del proyecto: contiene los archivos Java generados en base a los IDL y las implementaciones de las interfaces del *framework*.

Luego de realizar el *checkout* a una máquina local, se efectuaron las siguientes configuraciones:

- Configuración de datos de conexión hacia la base de datos en los siguientes archivos ubicados en el directorio "`etc/`":
  - `jdbc.properties`
  - `log4j.properties`
  - `scf.log4j.properties`
  - `app.log4j.properties`

  Los tres últimos corresponden a los archivos de configuración de log4j, librería de Java utilizada en los mensajes de logs escritos en tiempo de ejecución en la base de datos y mostrados a través del *prompt* del sistema operativo.
- Indicación en el archivo "`jacorb.properties`" (ubicado en el directorio "`etc/`") de la ubicación del archivo de logs "`NS_Ref`" utilizada por JacORB, el cual contiene una referencia inicial al servidor de nombres de CORBA (dado por un identificador IOR). Así el ORB desde el lado del cliente puede encontrar

el proceso del servidor de nombres, mediante una referencia conocida indicada en el archivo "`NS_Ref`".

- Ejecución del *script* "`environment.cmd`" ubicado en el directorio raíz del proyecto, el cual establece las variables de entorno requeridas.

5.2.4 Archivo *buid.xml* y preparación de la base de datos

Una vez realizadas las configuraciones indicadas en el punto anterior, se crearon las tablas en la base de datos utilizadas por el *framework*. Para esto, se empleó la herramienta Ant proporcionada por Apache. Teniendo Ant instalado y el la base de datos levantada, se ejecutó la siguiente línea de comandos en el directorio raíz del proyecto:

```
:\> ant prepare-db
```

La herramienta "Ant" busca en el archivo `build.xml` (ubicado en el directorio raíz del proyecto) el *target* "*prepare-db*", y ejecuta las instrucciones respectivas. Estas instrucciones corresponden a:

1. Ejecución del *script* "`ddl.sql`" ubicado en el directorio "`sql/`", el cual crea la tabla "logs" (utilizada para almacenar logs de actividad del *framework* mediante la herramienta log4j), y la tabla "service_types" (utilizada por el *framework* para verificar las SCF que están disponibles en el *gateway* de Parlay/OSA).
2. Ejecución del *script* "`input.sql`" ubicado en el directorio "`sql/`", el cual se encarga de ingresar valores en la tabla "service_types" correspondientes a algunos SCF de ejemplo. Esta tabla será utilizada posteriormente para registrar las tres SCFs utilizados por PAM (estas SCFs fueron descritas en los puntos 5.1.3.3, 5.1.3.4 y 5.1.3.5).

En la figura 33 es posible observar la ejecución del comando "`ant prepare-db`":

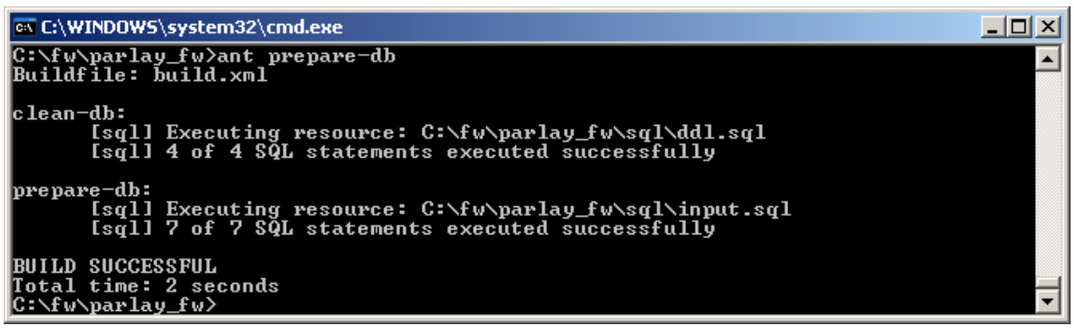


Figura 33. Ejecución del target “prepare-db” del archivo “*build.xml*”, utilizando Ant.

El archivo `build.xml` contiene varios *targets*, de los cuales, los más importantes se indican a continuación:

- “idl”: genera los archivos Java a partir de las especificaciones de interfaces del *framework* descritas a través de los archivos IDL correspondientes.
- “deploy”: ejecuta un *deploy* completo del proyecto compilando todas las clases Java que implementan el *framework*, y empaquetándolas en archivos jar los cuales son alojados posteriormente en el directorio “`dist/`”.
- “Javadoc”: genera toda la documentación Javadoc, la que es alojada luego en el directorio “`docs/`”.

5.2.5 Puesta en marcha del framework y del servidor de nombres

Una vez realizadas las configuraciones indicadas en el punto 5.2.3 y preparada la base de datos (punto 5.2.4), se procede con la ejecución del *framework* y del servidor de nombres utilizado para que las aplicaciones clientes puedan encontrar las referencias IOR a los objetos disponibles en el servidor a través del ORB.

Primero se debe levantar el servidor de nombres de CORBA. Para esto se ejecutó el archivo “`ns.bat`” ubicado en el directorio “`dist/framework/bin/ns/`”.

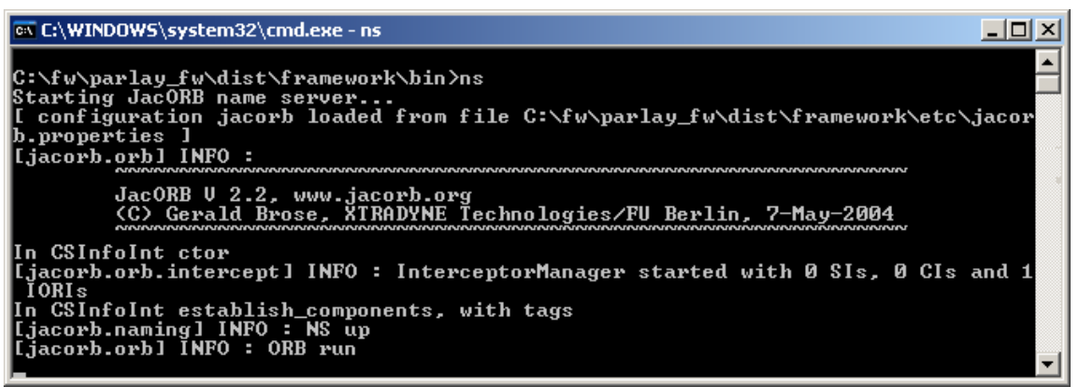


Figura 34. El servidor de nombres de CORBA en ejecución.

El servidor de nombres queda de esta manera en ejecución junto con el canal ORB abierto.

Levantado el servidor de nombres, la puesta en marcha del *framework* se realiza a través de la ejecución del *script* "`framework.bat`" ubicado en el directorio "`dist/framework/bin/ns/`".

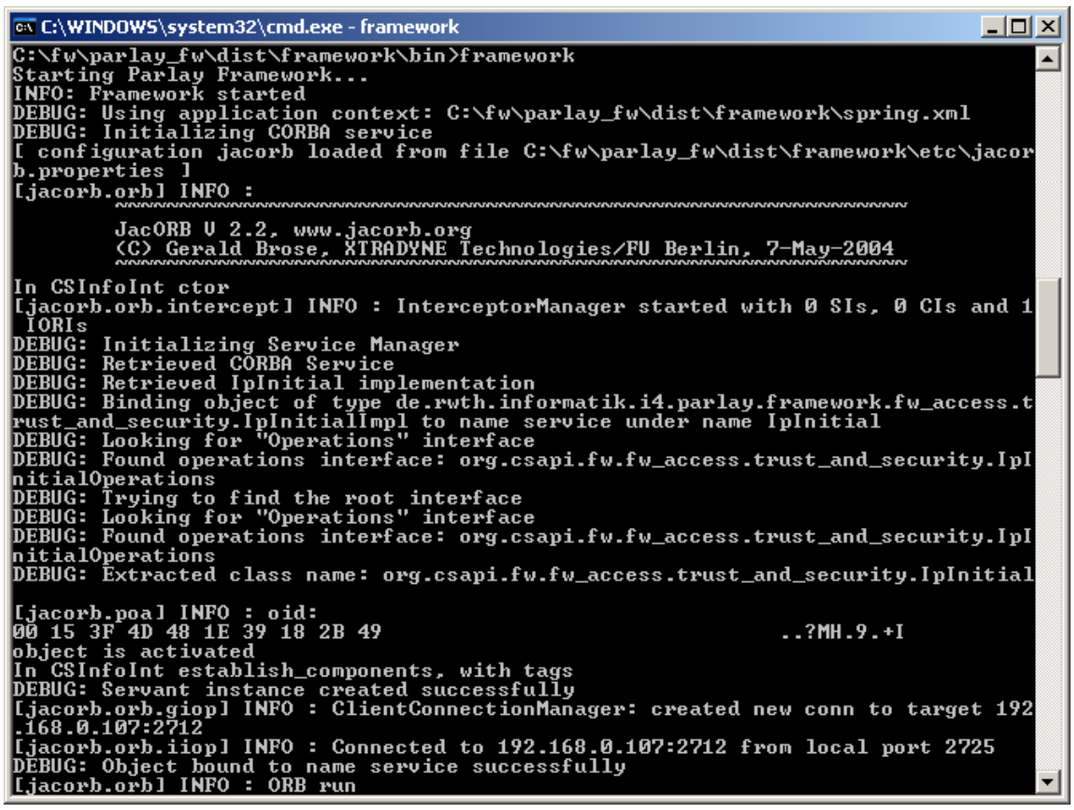


Figura 35. Puesta en marcha del framework de Parlay/OSA.

5.2.5.1 Clase IpInitial: interfaz principal del framework

Al comenzar la ejecución del *framework*, éste busca su interfaz principal denominada `IpInitial` e interactúa con el servidor de nombres para publicar en el ORB una referencia del objeto correspondiente a la interfaz anteriormente indicada, la que es utilizada luego por una aplicación cliente para iniciar la autenticación de ésta en la pasarela de Parlay/OSA. `IpInitial` también es utilizada por PAM para acceder a las otras interfaces del *framework*, con la finalidad de registrar en este último, las SCFs de PAM. La referenciación de esta interfaz en el servidor de nombres se puede observar en la siguiente figura:

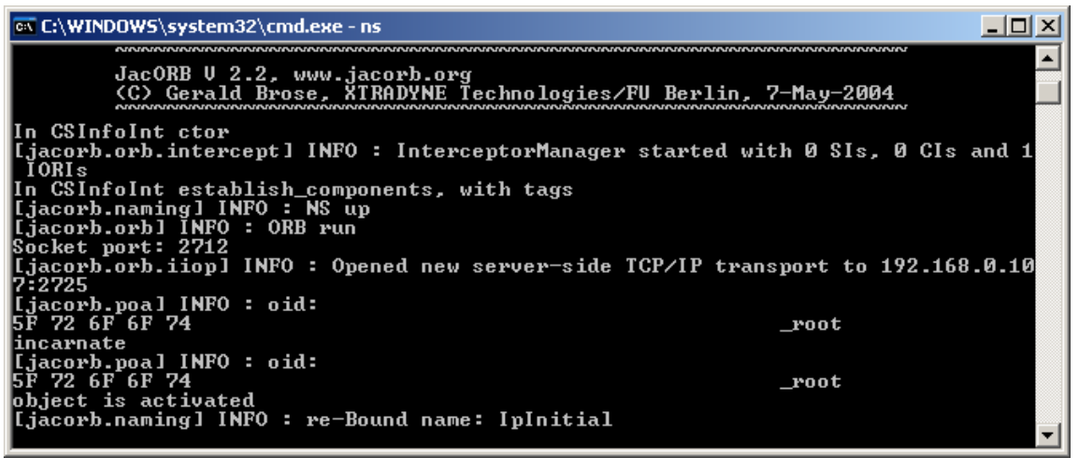


Figura 36. Referenciación de la interfaz del *framework* "`IpInitial`" en el servidor de nombres.

Desde este punto en adelante, el *framework* queda listo para llevar a cabo la interacción con las SCFs presentes en el *gateway* de Parlay/OSA, específicamente las SCFs implementadas en PAM, y para interactuar con las aplicaciones cliente que harán uso de estas capacidades de servicio.

5.3 Implementación del servicio PAM

5.3.1 Herramientas y tecnologías utilizadas

Como se señaló en el punto 5.2.2, en la implementación del *framework* se utilizó CORBA, y por lo tanto, en la implementación del servicio PAM también se hizo uso de esta tecnología, la cual hace posible que aplicaciones clientes puedan utilizar las capacidades de servicio de PAM bajo un entorno distribuido y mediante la utilización de un canal de comunicación ORB, en donde PAM pone a disposición sus interfaces en forma de objetos, pudiendo ser éstos invocados por las aplicaciones cliente, de forma remota y transparente en cuanto a la localización de estos objetos, todo lo anterior, bajo una previa autorización y validación de la aplicación cliente en el *framework* de Parlay/OSA (para revisar los aspectos y secuencias de interacción entre una aplicación cliente, el *framework* y una SCF, refiérase al punto 3.1.1.4).

Las herramientas y tecnologías involucradas en la implementación de PAM se listan a continuación:

- JacORB versión 2.2: implementación del estándar CORBA bajo Java, y compilador *IDL-to-Java* requerido para la generación de los archivos correspondientes (interfaces Java, clases *Holders* y *Helpers*, *Stubs* y *Skeletons*) a partir de los archivos IDL adjuntos en la especificación de Parlay/OSA, "parte 14" correspondiente a PAM [Ets05b].

- *Java 2 Standard Edition* versión 1.5: utilizado en la implementación de las interfaces Java de PAM generadas por el compilador JacORB.

- MySQL 5.0: utilizada para la persistencia de datos usados por PAM.

- Log4j: herramienta utilizada para realizar *debug* y muestra de logs en tiempo de ejecución del servicio PAM, a través del *prompt* del sistema operativo.

- Eclipse 3.1.1: entorno de desarrollo Java usado en la implementación de PAM. Se emplearon también los siguientes *plug-ins* para Eclipse:

    - Hibernate 3: *plug-in* requerido para realizar un mapeo Objeto-Relacional hacia la base de datos MySQL, desde las clases Java creadas para almacenar valores asociados a Agentes, Identidades y sus atributos (conceptos descritos en el punto 5.1.2), y mantener así la persistencia de estos valores.

    - Jigloo: Usado para construir la interfaces graficas en aplicaciones Java, específicamente en las aplicaciones que ponen a prueba el funcionamiento de las SCFs de PAM. Estas aplicaciones de prueba serán descritas al final del Capitulo V.

- Ant: herramienta libre desarrollada por la *Apache Software Fundation*, empleada para realizar tareas repetitivas y mecánicas, en fases de construcción y compilación (*deploy*) del proyecto PAM, tareas que son especificadas en un archivo XML con nombre `build.xml`.

- CVS: sistema de control de versiones y repositorio de archivos utilizado para mantener registro de todo el trabajo y los cambios en los archivos y códigos fuente del proyecto PAM.

5.3.2 Preparación del entorno de desarrollo y tareas preliminares

5.3.2.1 Creación del proyecto y estructura de directorios

El proyecto PAM fue creado bajo el nombre "`parlay_pam`" en el entorno de desarrollo Eclipse iniciando un nuevo proyecto Java, donde se crearon los siguientes directorios:

- `build/`: destinado para alojar todos los archivos compilados de java (archivos `.class`), a partir del código fuente del proyecto.
- `bin/`: alojamiento de archivos *scripts* `.bat` (para Windows) y `bash` (para Unix) creados para correr comandos de ejecución del compilador *IDL-to-Java*, ejecutar las SCFs de PAM, y *scripts* para ejecutar las aplicaciones clientes que se crearán para poner a prueba el servicio implementado.
- `dist/`: en este directorio se alojará el código Java distribuible, consistente en archivos `jar` que contienen las clases Java compiladas del proyecto.
- `etc/`: para situar archivos de configuración.
- `idl/`: para alojar los archivos IDL requeridos para crear las interfaces Java utilizadas en la implementación de PAM.
- `lib/`: directorio destinado para el alojamiento de todas las librerías `jar` utilizadas en la implementación. Estas librerías son incorporadas al proyecto creado en Eclipse.
- `sql/`: contiene *scripts* para crear una tabla de logs de actividad y otra para registrar las SCFs soportadas por el *framework*.
- `src/generated/`: contiene los archivos Java generados en base a los IDL.
- `src/main/`: contiene el código fuente Java correspondiente a la implementación del servicio PAM.

5.3.2.2 Archivos IDL requeridos

Los archivos IDL utilizados en la generación de las interfaces Java para implementar PAM fueron obtenidos desde el sitio oficial de Parlay[18], y se encuentran anexados a la documentación de especificación para la API de PAM, correspondiente a la parte 14: "*Presence and Availability Management SCF*" [Ets05b] de la especificación de Parlay/OSA 5.0 . Estos archivos son los siguientes:

- `pam_data.idl`: para la definición de los tipos de datos usados en PAM.
- `pam_interfaces.idl`: para la definición de las interfaces de PAM.

Además, fueron requeridos los IDL para los tipos de datos comunes a todas las SCF especificadas por Parlay/OSA, definidos en el archivo "`osa.idl`" (anexado a la parte 2: "*Common Data Definitions*" [Ets05d], de la especificación de Parlay/OSA 5.0), y los IDL que especifican las APIs del *framework* (anexados a la parte 3: "*Framework*" [Ets05c], de la especificación de Parlay/OSA 5.0) necesarios para realizar *casting* de algunas interfaces implementadas en el *framework*, las que son obtenidas por las SCFs de PAM a través del ORB de CORBA para interactuar con esté . El uso de estas clases y la conexión de las SCFs de PAM con el *framework* serán explicados en detalle en el punto 5.3.8.

Todos los archivos IDL utilizados fueron alojados en el directorio "`idl/`".

5.3.2.3 Creación del archivo build.xml y compilación IDL-to-Java

Con la finalidad de realizar tareas mecánicas y repetitivas tales como compilación de código en fase de desarrollo y organización de éste en determinados directorios, preparación de la base de datos mediante la ejecución de *scripts*, *deploy* del proyecto, entre otros procesos, se utilizó la herramienta Ant, la cual ejecuta las tareas que son especificadas a través de un archivo XML llamado `build.xml`, en donde cada tarea a realizar se especifica en un *target*. Para llevar a cabo una tarea determinada, en la línea de comandos se debe ejecutar el siguiente comando desde el directorio raíz del proyecto:

```
:\> ant <nombre_del_target>
```

[18] http://www.parlay.org

Durante el desarrollo del proyecto, este archivo siempre se mantuvo en construcción agregando nuevos *targets* a medida que iba surgiendo la necesidad de realizar nuevas tareas repetitivas.

A continuación, y a modo de ejemplo, se muestra la parte del archivo `build.xml` correspondiente al *target* "`idl`" creado para realizar la compilación *IDL-to-Java:*

```
<!-- Directory structure definitions -->
<property name="src.dir" location="src"/>
<property name="src.generated.dir" location="${src.dir}/generated"/>
<property name="idl.dir" location="idl"/>
<property name="idl.windows.script" location="${bin.dir}/idl.bat"/>
<property name="idl.unix.script" location="${bin.dir}/idl"/>

<!—- Targets -->
<target name="clean-idl">
  <delete dir="${src.generated.dir}"/>
  <mkdir dir="${src.generated.dir}"/>
</target>

<target name="idl" depends="clean-idl" description="Compiles IDL
files.">
  <exec executable="${idl.windows.script}" os="Windows XP">
    <arg line="-d ${src.generated.dir} ${idl.dir}/*.idl"/>
  </exec>
  <chmod file="${idl.unix.script}" perm="ugo+x"/>
  <exec executable="/bin/sh" os="SunOS">
      <arg line='-c "${idl.unix.script} -d ${src.generated.dir}
      ${idl.dir}/*.idl"'/>
  </exec>
</target>
```

Al ejecutar la instrucción "`:\> ant idl`", la herramienta Ant busca el *target* "idl" en el archivo `build.xml`, y ejecuta todas las instrucciones indicadas en el código mostrado anteriormente. Como se puede observar, existe dependencia entre *targets* indicada por la instrucción **`depends`**, por lo que al ejecutar el *target* "idl", primero se ejecuta el *target* "clean-idl" el cual elimina el directorio "`src/generated/`" (en caso que éste exista) y lo crea nuevamente. Luego, se lleva a cabo la ejecución del *script* "idl" (instrucción `<exec executable=…>`) alojado en el directorio "`bin/`". Se crearon *scripts* que pueden ser ejecutados en ambiente Windows (archivo `idl.bat`), y en ambiente Unix (archivo `idl`). Ant detecta automáticamente el sistema operativo sobre el cual se trabaja. En la instrucción `<arg line=…>` se indican los argumentos entregados al *script*, correspondientes a la ubicación de los archivos idl a compilar, y al directorio destino de los archivos Java generados.

La ejecución del *target* “idl” bajo ambiente Windows se puede observar en la figura 37:

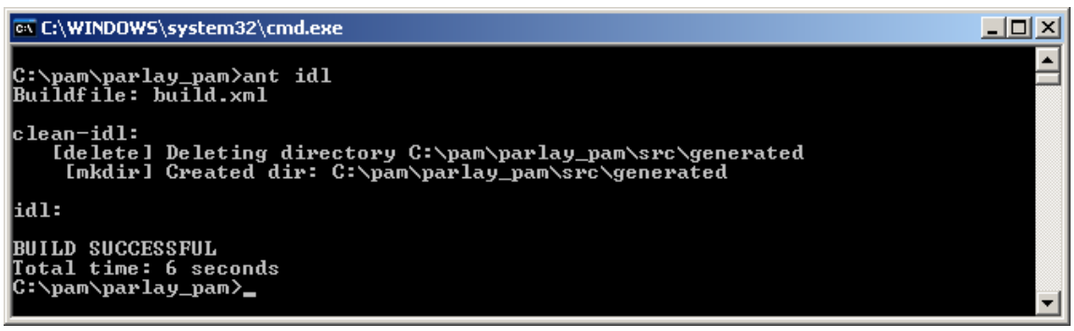


Figura 37. Compilación IDL-to-Java realizada con la herramienta Ant.

Los principales *targets* creados en el archivo `build.xml`, sin considerar los *targets* dependientes, se indican a continuación:

- **“add-services”**: efectúa la ejecución de un *script* sql que agrega valores a la tabla “service_types” correspondientes a las tres SCF de PAM, para que el *framework* pueda saber que éstas se encuentran implementadas en el *gateway* de Parlay/OSA. Estos valores corresponden a los *strings* “`P_PAM_PROVISIONING`”, “`P_PAM_ACCESS`” y “`P_PAM_EVENT`”.
- **“deploy”**: compila el código fuente situado en el directorio “`src/`” y lo deja en el directorio “`build/`” (tarea realizada por el *target* dependiente “compile”), y luego empaqueta en archivos `jar` todo lo compilado generando los archivos distribuibles del proyecto, alojándolos en el directorio “`dist/`”.

5.3.2.4 Archivos de configuración

Se crearon los siguientes archivos de configuración, los que fueron situados en el directorio “`etc/`”:

- `jdbc.properties`: datos de conexión a la base de datos, utilizados en el archivo `build.xml` para llevar a cabo tareas de ejecución de *scripts* sql.
- `log4j.properties`
- `scf.log4j.properties`
- `app.log4j.properties`

Los tres últimos archivos son empleados por la herramienta log4j, librería de Java utilizada para registrar en la base de datos mensajes de actividad de PAM y de las aplicaciones clientes que hacen uso de las SCFs de PAM, mientras se está en tiempo de ejecución. En estos archivos de configuración, se indican los datos de conexión a la base de datos, y las tablas y campos en donde se escribirán los logs de actividad.

5.3.3 Interfaces y clases Java generadas a partir de los archivos IDL

Tras la compilación *IDL-to-Java*, se generaron todas interfaces Java a implementar en PAM, además de las clases *Holders* y *Helpers*, *Stubs* y *Skeletons* respectivas a cada interfaz, utilizadas para la comunicación distribuida de objetos (interfaces PAM) bajo CORBA.

Las clases de interfaces generadas son organizadas en un esquema jerárquico de paquetes, en donde el nombre del paquete raíz corresponde a "`org.csapi`".

La figura 38 muestra esta jerarquía de paquetes:

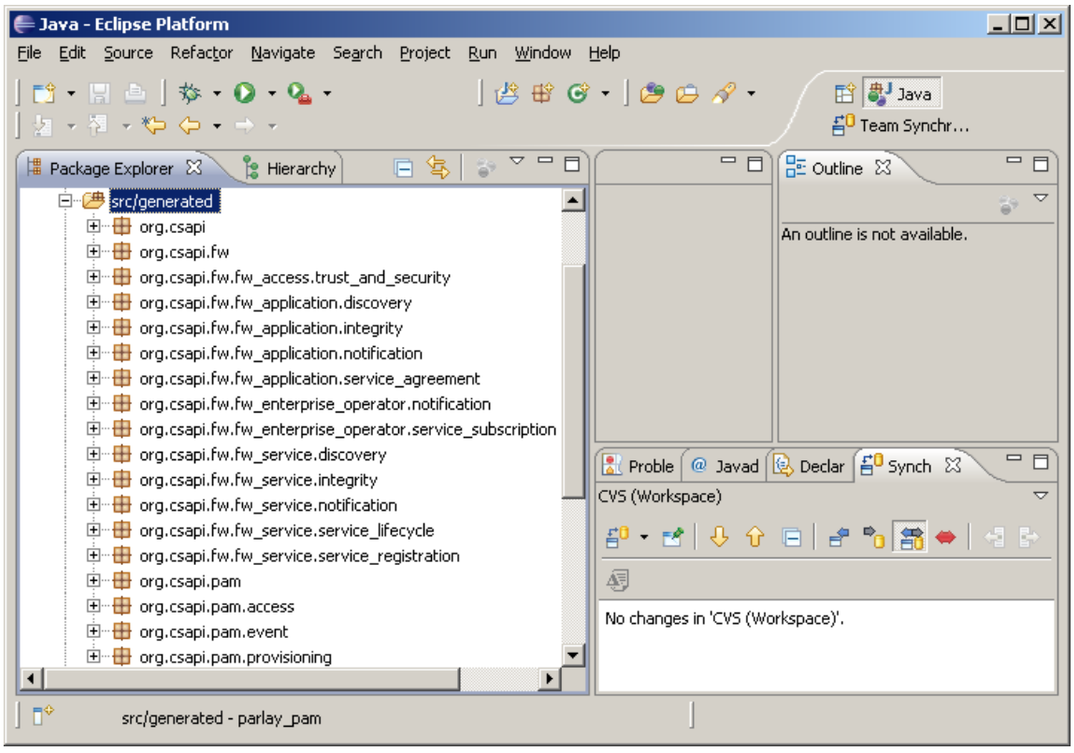


Figura 38. Jerarquía en el esquema de paquetes de clases generadas.

El contenido de cada paquete se describe a través de la siguiente tabla:

Tabla 4. Jerarquía de paquetes de clases generadas desde los IDLs, y su contenido.

Jerarquía de paquetes	**Contenido**
`org.csapi`	- Tipos de datos comunes - Excepciones comunes
`org.csapi.fw`	- Tipos de datos para el *framework* - Excepciones para el framework
`org.csapi.fw.*`	- Paquetes de interfaces del *framework* - Paquetes de tipos de datos para cada interfaz del fw - Paquetes de excepciones para cada interfaz del fw
`org.csapi.pam`	- Tipos de datos para PAM - Excepciones para PAM
`org.csapi.pam.access`	- Interfaces para la SCF PAM *Access*
`org.csapi.pam.event`	- Interfaces para la SCF PAM *Event*
`org.csapi.pam.provisioning`	- Interfaces para la SCF PAM *Provisioning*

Tal como se indicó en algún momento, de todas las clases incluidas correspondientes a las APIs del *framework* que fueron incorporadas en el proyecto, sólo algunas son utilizadas en la implementación de PAM y en la implementación de las aplicaciones de prueba, con la finalidad de hacer *casting* a los objetos o interfaces del *framework* obtenidas a través del ORB de CORBA, las que son finalmente utilizadas por PAM y por las aplicaciones de prueba para invocar métodos remotos en el *framework* llevando a cabo los procesos de autenticación de PAM y de las aplicaciones clientes, y de descubrimiento de las SCFs presentes en la pasarela, procesos que fueron descritos en el punto 3.1.1.4.

Las clases de interfaces generadas para las tres capacidades de servicio que componen a PAM junto con sus métodos se encuentran diseñadas bajo diagramas de clases UML, los cuales fueron mostrados oportunamente en el punto 5.1.3.

El listado y descripción de tipos de datos comunes y excepciones utilizadas en la implementación de PAM y del *framework*, queda fuera del alcance del presente trabajo, debido a la gran cantidad de tipos de datos y excepciones existentes, y a que éstos se encuentran debidamente descritos en los documentos de especificación de Parlay/OSA 5.0: parte 2: "*Common Data Definitions*" [Ets05d], parte 3: "*Framework*" [Ets05b] y parte 14: "*Presence and Availability Management SCF*" [Ets05c].

5.3.4 Esquema jerárquico de paquetes: organización de clases e interfaces implementadas

Las clases e interfaces implementadas fueron organizadas en un esquema jerárquico de paquetes, en donde el nombre del paquete raíz corresponde a "`de.rwth.informatik.i4.parlay`", nombre inspirado en el lugar de desarrollo del proyecto, correspondiente al Instituto 4 de Informática de la Universidad RWTH-Aachen.

El directorio "`src/main/`" fue destinado para contener el código fuente de las clases e interfaces implementadas.

El esquema jerárquico de paquetes se muestra en la figura 39:

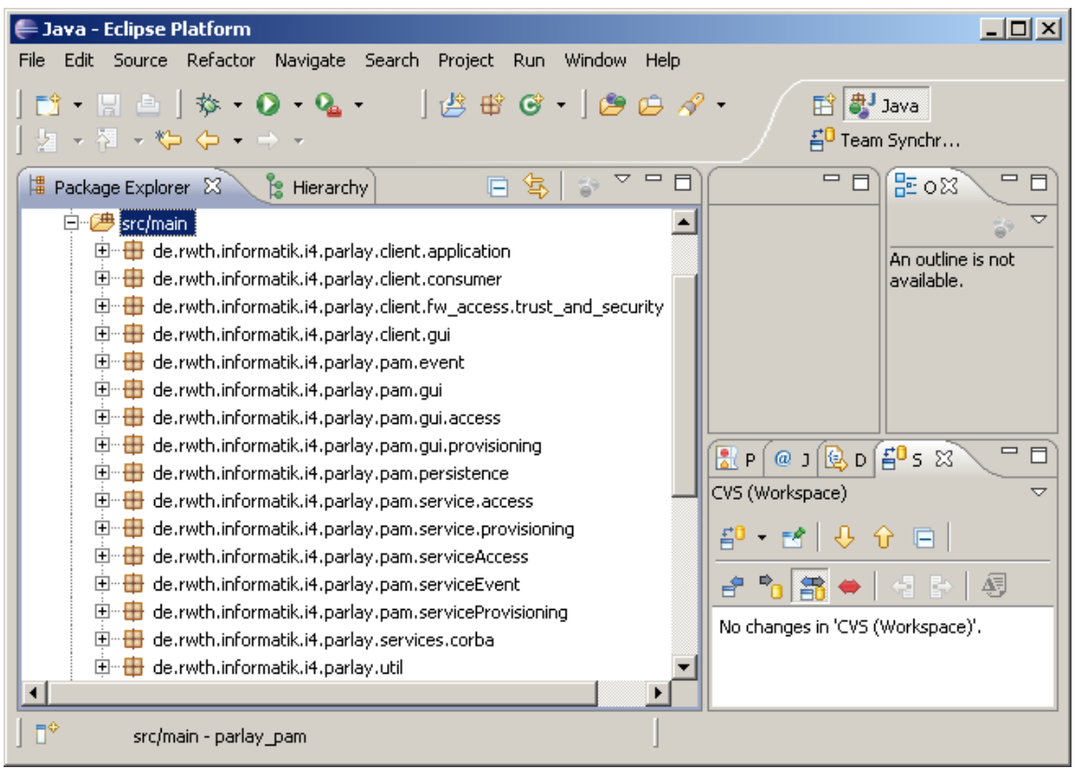


Figura 39. Esquema de paquetes de clases e interfaces implementadas en el proyecto PAM.

El contenido de cada paquete se describe en la tabla 5:

Tabla 5. Jerarquía de paquetes de clases de interfaces implementadas en PAM.

Jerarquía de paquetes	**Contenido**
…i4.parlay.client.* …i4.parlay.pam.gui …i4.parlay.pam.gui.access …i4.parlay.pam.gui.provisioning	- Clases de la aplicación cliente creada para poner a prueba la funcionalidad de las SCFs de PAM.
…i4.parlay.pam.presistence	- Clases utilizadas para realizar la persistencia de datos utilizados en el servicio PAM, a través de un mapeo Objeto-Relacional llevado a cabo con la herramienta Hibernate.
…i4.parlay.pam.serviceAccess	- Clases creadas para poner en marcha la SCF PAM *Access*
…i4.parlay.pam.serviceEvent	- Clases creadas para poner en marcha la SCF PAM *Event*
…i4.parlay.pam.serviceProvisioning	- Clases creadas para poner en marcha la SCF PAM *Provisioning*
…i4.parlay.pam.access	- Interfaces implementadas para la SCF PAM *Access*
…i4.parlay.pam.event	- Interfaces implemetandas para la SCF PAM *Event*
…i4.parlay.pam.provisioning	- Interfaces implementadas para la SCF PAM *Provisioning*
…i4.parlay.services.corba	- Clases que implementan todas las funcionalidades requeridas para la comunicación cliente-servidor en CORBA, ocultando la complejidad de éste estandar, mediante un uso simple de los métodos de estas clases por parte las demás clases e interfaces implementadas en PAM y que requieren realizar peticiones a través del canal de comunicación ORB.
…i4.parlay.pam.util	- Clases con algoritmos de criptografia utilizadas por el *framework*, por las aplicaciones clientes y por las SCF para realizar una comunicación segura entre estos tres elementos.

5.3.5 CorbaServiceImpl.java: implementación de métodos CORBA usando JacORB.

Con la finalidad de enfocarse en la implementación del estándar Parlay/OSA, más que en la complejidad y mecanismos de comunicación distribuida de CORBA, se utilizó una clase llamada `CorbaServiceImpl.java`, la cual implementa todas las funcionalidades requeridas para llevar cabo, mediante una serie de métodos, la

comunicación bajo CORBA entre PAM, el *framework* y las aplicaciones clientes. Los principales métodos de esta clase y sus descripciones respectivas se indican a continuación:

- Método `org.omg.CORBA.Object` **`connect`**`(Object obj)`: conecta al ORB el objeto entregado como parámetro, y devuelve la instancia del objeto *servant* respectivo.
- Método void **`bind`**`(String name, Object obj)`: vincula en el servidor de nombres el objeto entregado con el nombre indicado.
- Método `org.omg.CORBA.Object` **`resolve`**`(String name, Class clazz)`: recupera un objeto pasando como parámetro su clase interfaz respectiva y el nombre con el que se publicó este objeto en el servidor de nombres.
- Método `org.omg.CORBA.Object` **`narrow`**`(org.omg.CORBA.Object obj, Class clazz)`: realiza un *casting* o conversión de tipos de objetos pasados como parámetro, desde un objeto de tipo genérico “obj”, hacia el tipo de objeto “clazz”.
- Método `void` **`deactivate`**`(org.omg.CORBA.Object obj)`: inicia la desactivación del objeto pasado como parámetro. La desactivación finaliza cuando todas las peticiones activas sobre el *servant* terminan.
- Método `public byte[]` **`getReference`**`(org.omg.CORBA.Object obj)`: obtiene una referencia del objeto pasado como parámetro. Esta referencia consiste en un identificador único para éste.
- Método `public org.omg.CORBA.Object` **`getObject`**`(byte[] b)`: retorna el objeto correspondiente a la referencia de éste que se pasa como parámetro.
- Método `public ORB` **`getORB`**`()`: retorna la referencia del ORB en uso.
- Método `public static synchronized CorbaService` **`getInstance`**`()`: retorna una instancia de la clase `CorbaServiceImpl` para poder hacer uso de sus métodos.

5.3.6 Interfaces PAM implementadas

En el punto 5.1.3 se describió la función de todas las interfaces que componen la capacidad de servicio PAM, integrada por tres SCFs: **PAM *Provisioning*, PAM *Access* y PAM *Event***. Las interfaces que componen estas tres capacidades de servicios son organizadas en paquetes de clases, un paquete para cada SCF de PAM (destacados en negrita en la tabla 5). Además de la función que cumple cada interfaz, mediante la representación de diagramas de clases UML se señalaron los métodos pertenecientes a cada clase y sus relaciones. La descripción, parámetros, y valores de retorno de cada método, se encuentran descritos en forma detallada en el documento de especificación de Parlay/OSA correspondiente a la parte 14: "*Presence and Availability Management*" [Ets05c].

Los tres SCFs anteriormente señalados componen en su conjunto la capacidad de servicio de PAM. El presente trabajo se concentra principalmente en la implementación de las interfaces de PAM.

La próxima figura muestra la jerarquía de paquetes creada para el proyecto PAM, y se señalan en color verde las clases correspondientes a las implementaciones de las interfaces para cada SCF, las que siguen un esquema de nombres con el siguiente formato: `<nombre_de_la_interfaz>Impl.java`:

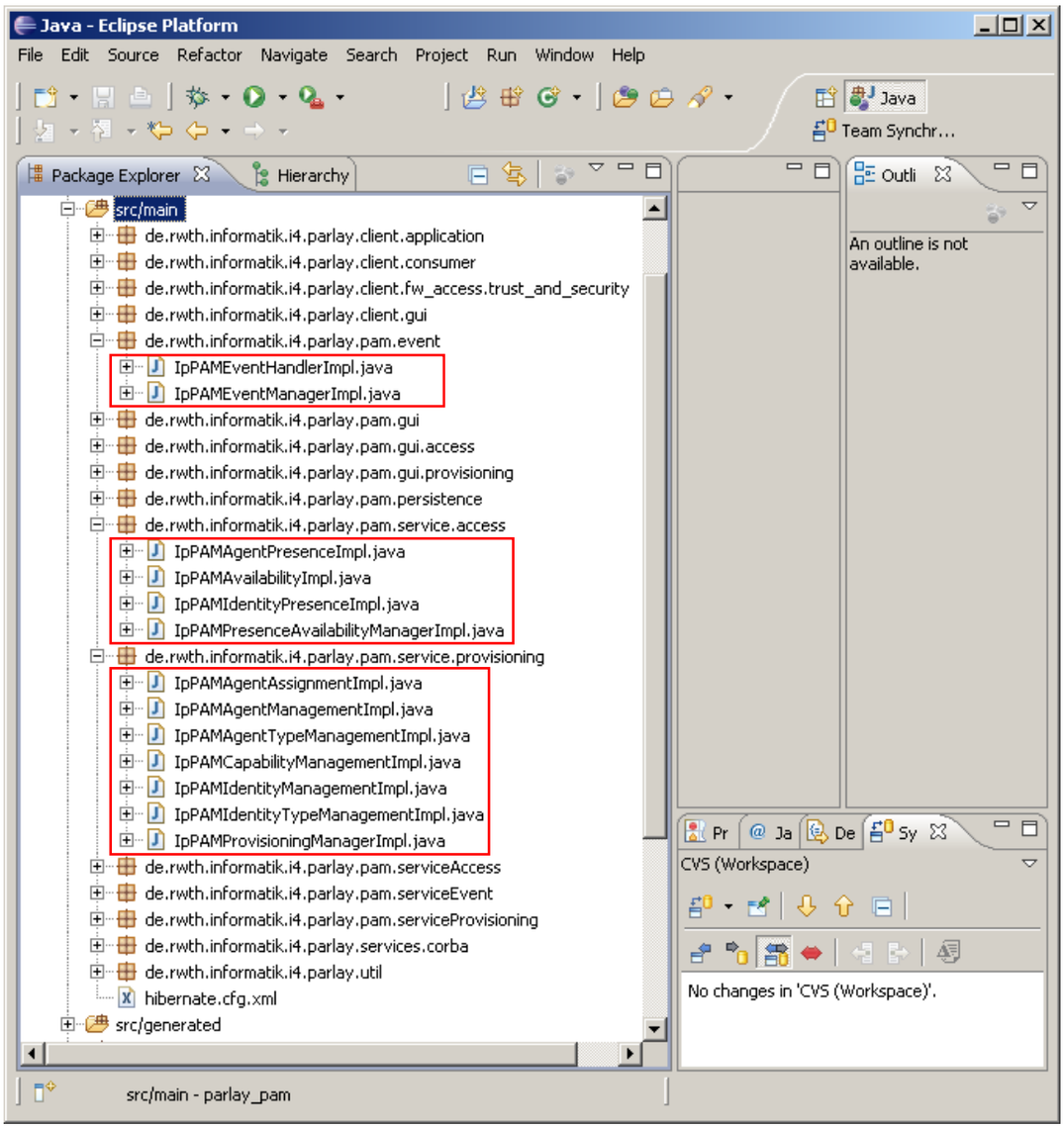


Figura 40. Esquema de paquetes de las clases implementadas creado para el proyecto PAM.

5.3.6.1 Procesamiento de información de presencia y disponibilidad

La información de presencia es representada y almacenada a través de atributos definidos para un agente determinado, por ejemplo, ubicación de un agente, *power status*, o para una capacidad especifica de un agente, como por ejemplo, estado de presencia del agente para voz/mensajería.

La información de presencia de una identidad, sigue la misma lógica de representación de información mediante atributos. Por ejemplo, un atributo asociado a un tipo de identidad e identificado con el nombre "`P_SUBSCRIBER_STATUS`", podría tomar el valor "`En reunión`".

La información relacionada con “disponibilidad” de una identidad, administrada a través de la interfaz `IpPAMAvailabilityImpl` perteneciente a la SCF Access, es procesada verificando la disponibilidad para una determinada capacidad o medio de comunicación (SMS, voz, mensajería instantánea, etc.) asociado a la identidad. Esta verificación sigue el siguiente algoritmo:

Paso 1: Se verifica la disponibilidad de la identidad consultando sus atributos creados para almacenar información de disponibilidad.

Paso 2: Se verifican los agentes que soporten la capacidad entregada como parámetro y que tengan esta capacidad disponible como medio de comunicación.

Paso 3: Se retorna un arreglo con la siguiente información para cada capacidad disponible como medio de comunicación:

- Nombre (identificador) de la identidad
- Estado del suscriptor: información obtenida en el 1er paso.
- Medio de comunicación: nombre de la capacidad, ej.: IM, MMS, VOICE, SMS.
- Dirección del contacto: contiene el nombre del agente que posee la capacidad indicada en el campo “Medio de comunicación”.

De esta manera, una aplicación cliente podría consultar el estado de disponibilidad de un suscriptor (identidad) determinado, obteniendo la información descrita en el paso 3. Una identidad estará disponible sólo si el campo “Estado del suscriptor” tiene un valor de “disponible” y existe alguna capacidad disponible como medio de comunicación.

5.3.7 Persistencia de datos: mapeo Objeto-Relacional usando la herramienta Hibernate

La administración de identidades y agentes, y toda la información referente a sus atributos y capacidades (SMS, voz, mensajería instantánea, etc.), información de presencia y disponibilidad de éstos, es manejada por las capacidades de servicio PAM *Access* y PAM *Provisioning*, tal como se describió en el punto 5.1.3.1. Toda esta información debe ser almacenada en la base de datos, ya sea para mantenerla persistente en el tiempo y realizar actualizaciones, como por ejemplo, información relacionada con el conjunto de agentes e identidades (y sus relaciones) creadas en el sistema, junto con

sus atributos y capacidades, o para almacenar información temporal de presencia y disponibilidad, para que ésta pueda ser consultada durante el tiempo de sesión de una aplicación cliente.

Utilizando el paradigma orientado a objetos de Java, agentes, identidades y capacidades, se representaron mediante clases con sus atributos respectivos, manipulados a través de métodos `get` y `set`. Así, por ejemplo, un agente queda representado mediante la siguiente clase:

```
public class Agent  {
	private String agentName;
	private Long id;
	private Set capabilityList = new HashSet();
	private Set agentTypeList = new HashSet();
	private Set<Identity> users = new HashSet() ;

	public String getAgentName() {
		return agentName;
	}
	public void setAgentName(String agentName) {
		this.agentName = agentName;
	}
	/*... métodos get y set restantes para cada atributo */
}
```

Todas las clases creadas para llevar a cabo la persistencia de datos, fueron alojadas dentro del paquete `de.rwth.informatik.i4.parlay.pam.persistence`. Estas clases se listan en la figura 41:

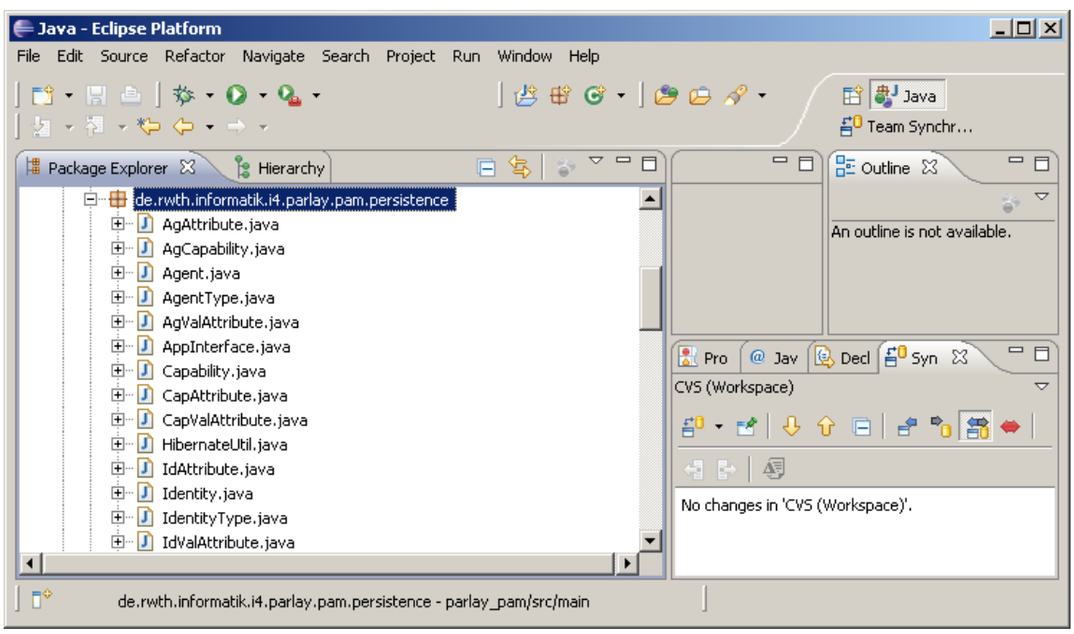


Figura 41. Lista de clases creadas para mantener la persistencia de datos.

De esta forma, en la implementación de las interfaces de PAM *Provisioning*, agentes, identidades, sus relaciones y capacidades son representados mediante clases Java, las que se pueden crear con una simple llamada al operador `new`, para luego, mediante los métodos `get` y `set` obtener y actualizar los valores para cada atributo de la nueva instancia creada. En las interfaces de PAM Access, se utilizó el mismo principio, agregando/actualizando, mediante métodos `set`, y obteniendo, mediante métodos `get`, información de disponibilidad y presencia de agentes e identidades.

En Java, la creación de instancias y la asignación de sus atributos, se mantiene sólo en tiempo de ejecución, siendo necesario almacenar esta información. Es aquí donde entra en juego la herramienta Hibernate[19], la cual permite realizar un mapeo Objeto-Relacional (OR), es decir, almacenar en una base de datos relacional todos los datos manipulados por objetos o clases Java en tiempo de ejecución, junto con la creación y destrucción de éstos.

Para realizar el mapeo OR, se implementó en la base de datos MySQL el siguiente modelo relacional, el cual fue diseñado en base a las clases creadas para la persistencia (listadas en la figura 41) y sus relaciones. Un ejemplo de relación sería la posibilidad de asignar un agente a una identidad. El modelo relacional creado se muestra en la figura 42:

[19] Sitio official: http://www.hibernate.org/

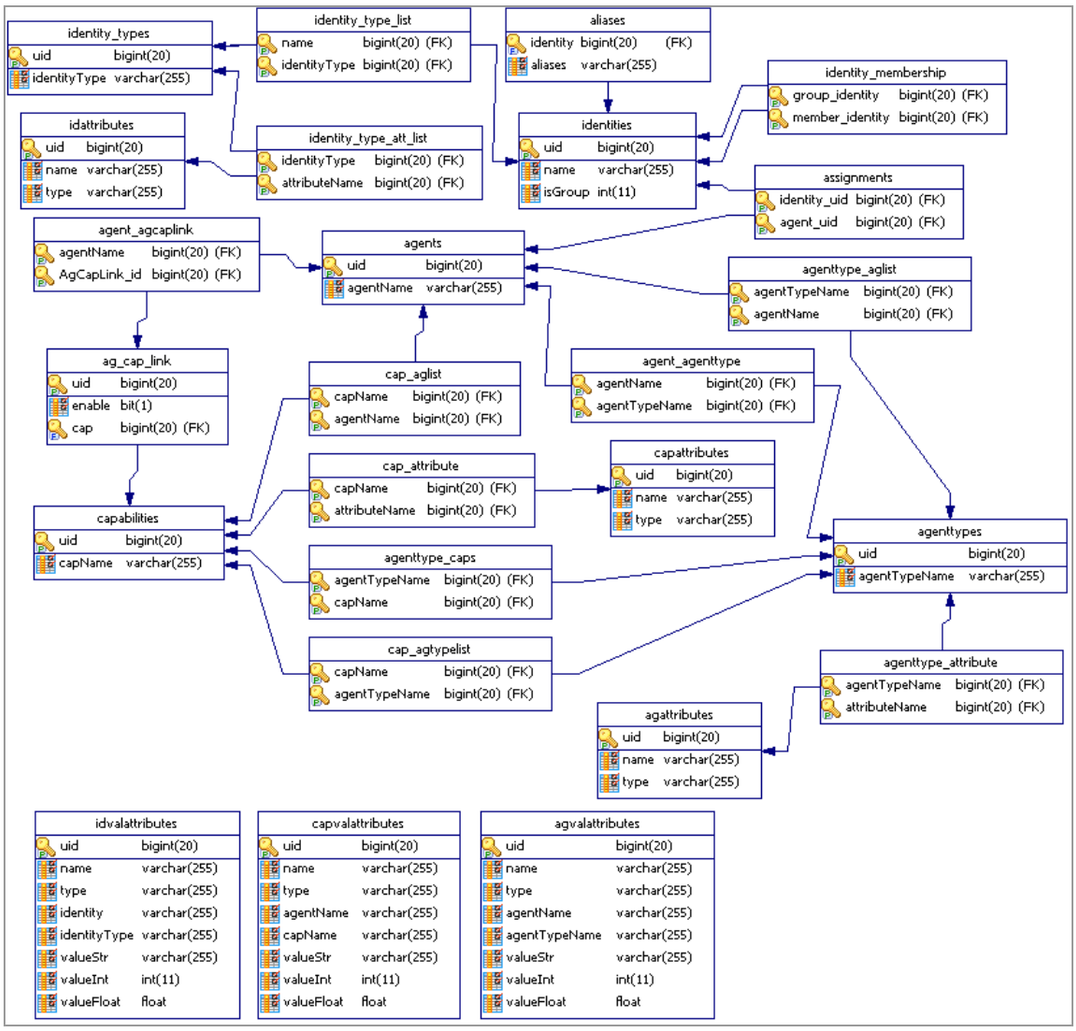


Figura 42. Modelo relacional utilizado en el mapeo Objeto-Relacional.

Con Hibernate no se requiere incrustar código SQL en la implementación, la herramienta posee su propio lenguaje para realizar el mapeo OR llamado HQL (*Hibernate Query Language*), sin embargo alternativamente el manejo de datos también puede efectuarse mediante consultas nativas SQL. El mapeo OR queda especificado a través de archivos declarativos XML que permiten establecer las relaciones entre una clase con sus atributos, y las tablas y campos respectivas en la base de datos.

5.3.8 Interacción PAM-Framework: Interfaces implementadas

Retomando los pasos requeridos para la interacción entre una capacidad de servicio y el *framework*, descritos en el punto 3.1.1.4, y representados gráficamente en

la figura número 15, se realizará a continuación una analogía entre la interacción PAM-*Framework* y las interfaces implementadas que fueron usadas en las clases principales de PAM *Provisioning*, PAM *Event* y PAM *Access* creadas para poner en marcha estos servicios.

Se crearon tres clases principales (que contienen el método `main` de Java) bajo el nombre "`ServiceMain.java`", una para cada SCF de PAM (ver fig.43). El código escrito en estas tres clases es prácticamente el mismo, dado que, independiente del SCF que se esté poniendo en marcha, se utilizan los mismos mecanismos de interacción SCF-*Framework* y es por este motivo que para explicar la implementación de los pasos de interacción del SCF con el *framework*, se tomará como referencia sólo una clase "`ServiceMain.java`".

Estas tres clases principales fueron añadidas en los paquetes respectivos para cada SCF:

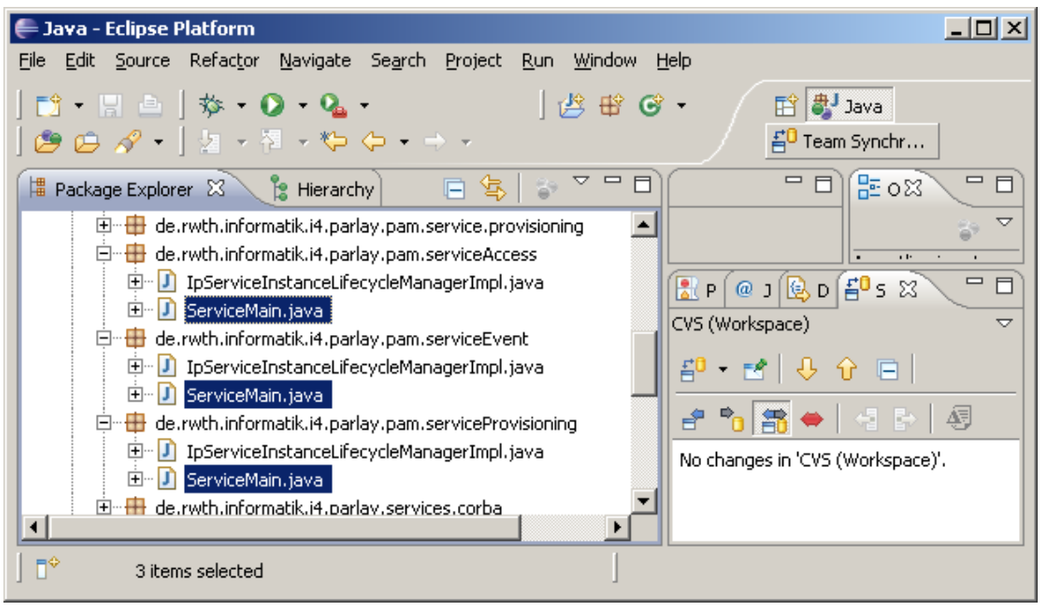


Figura 43. Paquetes que contienen la clase *ServiceMain.java*

La interacción *Framework*-Servicio(SCF), que fue descrita en la fundamentación teórica en la figura 15, la cual citada nuevamente más abajo en la Figura 44 para una mejor explicación, se lleva a cabo en cinco pasos: 1:autenticación , 2:solicitud de interfaz de registro, 3:registro de la factoría o servicio, 8:creación de un administrador de servicio, 9:entrega al *framework* de este administrador de servicio. Es importante señalar que para realizar los dos últimos pasos (8 y 9), no es necesario esperar que una aplicación cliente realice los pasos previos de autenticación,

descubrimiento y solicitud de uso de un servicio (pasos 4, 5, 6 y 7), por consiguiente, los pasos 8 y 9 pueden realizarse inmediatamente después de los pasos 1, 2 y 3.

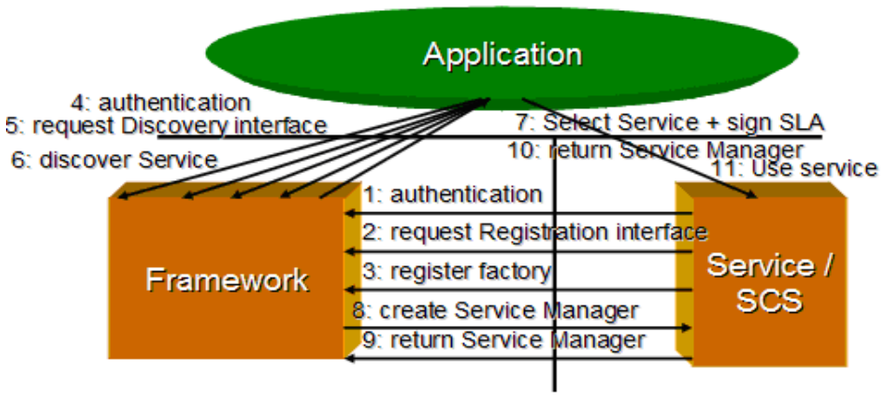


Figura 44. Autenticación y registro de servicios (SCFs) en el *framework*. Fuente [Moe02].

La autenticación realizada por una SCF sigue los mismos procedimientos que la autenticación de una aplicación cliente.

Para realizar las tareas de comunicación de PAM con el *framework* a través de CORBA se utilizó la clase `CorbaServiceImpl` (descrita en el punto 5.3.5).

Cada paso, junto con las clases e interfaces utilizadas en "`ServiceMain.java`" se describen a continuación:

Pasos previos: búsqueda de la interfaz principal del *framework* `IpInitial`: Se obtiene una referencia de la interfaz principal del *framework* para poder invocar de forma remota los métodos implementados en ésta.

1:Autenticación: Proceso mediante el cual el *framework* autentica a la SCF (o eventualmente, a una aplicación cliente). Opcionalmente, este proceso puede ser realizado también para que una SCF (o una aplicación cliente) autentique al *framework* que desea utilizar. Para iniciar la autenticación con el *framework*, la SCF invoca el método `initiateAuthenticationWithVersion()`de `IpInitial` (implementada en el *framework*). Los parámetros pasados en este método corresponden a:

a. Un "domino de servicio (o de cliente)" determinado por: un *string* del tipo "`rwth.de/service<nroServicio>`" (o en caso de una aplicación cliente, "`rwth.de/client<`*`nroCliente`*`>`") que identificada de manera única al servicio (o cliente), y por una instancia de la clase `IpClientAPILevelAuthentication` (implementada en PAM), la que contiene algoritmos para realizar una autenticación por desafío mutuo basado en la

verificación periódica de la identidad del SCF, usando intercambio de información encriptada.

b. El tipo de mecanismo de autenticación requerido por el SCF, el que queda determinado por un *string*, en este caso con el valor "`P_OSA_AUTHENTICATION`", correspondiente al nivel de mecanismo de la API OSA. La alternativa es usar el mecanismo proporcionado a nivel CORBA.

c. El identificador de la versión del *framework* que el SCF solicita. En este caso, este identificador queda determinado por un *string* con valor "`SP_PARLAY_5`", correspondiente a la versión 5.0 de la especificación Parlay/OSA para el *framework*.

El método `initiateAuthenticationWithVersion()` al ser invocado retorna una referencia a la interfaz `IpAPILevelAuthentication` (implementada en el *framework*) utilizada por la SCF para dar inicio al proceso de autenticación a través de una llamada del método `challenge()`, comenzando de esta manera, la verificación de la autenticidad del *framework*. Realizada la verificación, se invoca al método `authenticationSucceeded()` de `IpAPILevelAuthentication` para informar finalmente al *framework* que se ha completado exitosamente la verificación de autenticidad de éste, esperando luego que el *framework* confirme este proceso. La espera por parte de la SCF se realiza invocando a `IpClientAPILevelAuthenticationImpl.wait()`.

2:Solicitud de la interfaz de registro: Para obtener la interfaz de registro de las SCFs implementada en el *framework*, primero se obtiene una referencia de la interfaz `IpAccess` (implementada en el *framework*) llamando al método `requestAccess()`de `IpAPILevelAuthentication`, para luego invocar al método `obtainInterface("P_REGISTRATION")` de `IpAccess`, el que finalmente retorna una referencia a la interfaz de registro de servicios `IpFwServiceRegistration` (implementada en el *framework*) .

3:Registro de la factoría o SCF: con la referencia a la interfaz `IpFwServiceRegistration`, el registro de un SCF en el *framework* se realiza llamando al método `registerService()` pasando como parámetro la SCF que se desea registrar. Este parámetro consiste en un *string* y cada SCF especificado en Parlay/OSA tiene un valor para este *string*, que en el caso de las SCFs de PAM corresponde a

“P_PAM_ACCESS”, “P_PAM_PROVISIONING” o “P_PAM_EVENT”, según la SCF que se esté registrando. Un identificador de servicio finalmente es retornado.

8:Creación de un administrador de servicio: Para cada SCF de PAM se implementó una interfaz IpServiceInstanceLifecycleManager, la cual tiene la función de permitir al *framework* tener acceso a esta interfaz de administración del SCF que se registró previamente, con la finalidad de que el *framework* pueda retornar a las aplicaciones cliente una referencia a esta interfaz para que puedan hacer uso de la interfaces implementadas en el servicio registrado.

La interfaz IpServiceInstanceLifecycleManager es el primer punto de contacto entre la aplicación cliente y la SCF que solicita.

9:Entrega al framework del administrador de servicio: El último paso de interacción PAM-*Framework* es el traspaso al *framework* de la referencia de la clase IpServiceInstanceLifecycleManagerImpl (correspondiente a la implementación de la interfaz IpServiceInstanceLifecycleManager). Este traspaso se realiza invocando al método announceServiceAvailability() de la interfaz de registro IpFwServiceRegistration obtenida en el paso 2, y pasando como parámetros la referencia de IpServiceInstanceLifecycleManagerImpl, y el identificador de servicio obtenido al final del paso 3.

Finalmente, la última sentencia ejecutada en ServiceMain.java corresponde a la llamada al método run()del ORB en uso. De esta manera, el SCF registrado queda a la espera de que alguna aplicación cliente requiera hacer uso de sus interfaces.

5.3.9 Preparación del entorno de ejecución del servicio PAM

5.3.9.1 Deploy de PAM usando la herramienta Ant

El *deploy* del proyecto consiste en diversos pasos: el primero es compilar el código fuente implementado ubicado en “scr/” y dejarlo en el directorio “build/”. Luego se construyen los archivos ejecutables .jar para cada SCF de PAM, los que son destinados al directorio “dist/” junto con las librerías ubicadas en “lib/” requeridas para la ejecución de éstos, los archivos de configuración properties ubicados en “etc/”, y los *scripts* de ejecución de los .jar creados, en el directorio “bin/”.

Resulta de esta manera, una versión distribuible del proyecto PAM, la que es alojada finalmente en el directorio "`dist/`".

Todos estos pasos fueron especificados en el *target* "deploy" del archivo `build.xml` (descrito en el punto 5.3.2.3), por lo que para ejecutarlos, se requiere tan sólo de llamar al siguiente comando estando ubicado en el directorio raíz del proyecto (ruta de ubicación del archivo *`build.xml`*):

```
:\> ant deploy
```

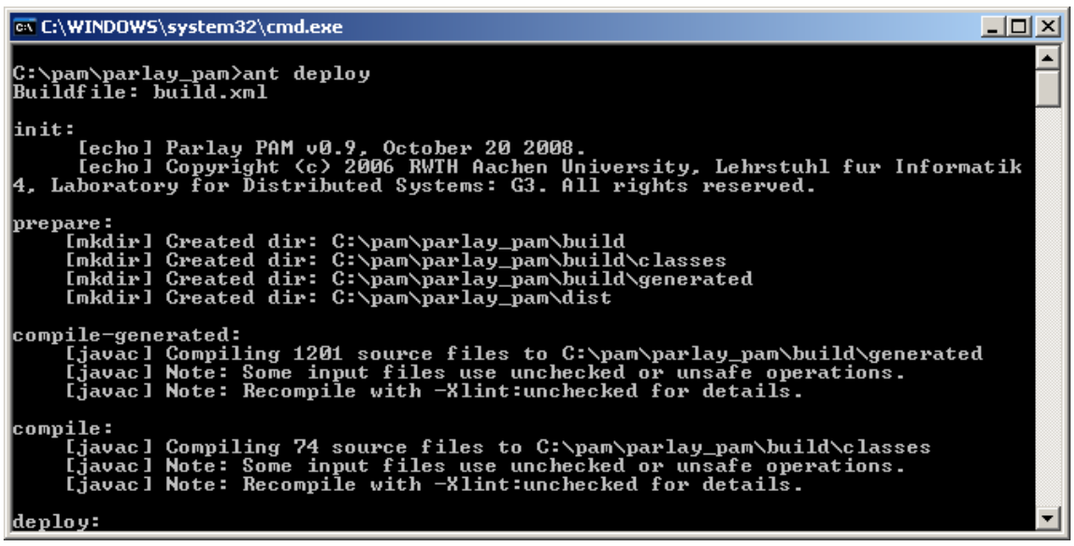


Figura 45. Deploy de PAM usando la herramienta ANT. Se muestran sólo los primeros pasos realizados después de ejecutar "ant deploy".

5.3.9.2 Registro de los SCFs en la base de datos

El *framework* verifica qué servicios que se encuentran implementados en el *Gateway* de Parlay/OSA, a través de una consulta a la base de datos, específicamente revisando los valores (*Strings*) presentes en la tabla "service_types". Las SCFs de PAM son identificadas por los valores "`P_PAM_PROVISIONING`", "`P_PAM_ACCESS`" y "`P_PAM_EVENT`", los que deben ser agregados a la tabla indicada anteriormente. Para realizar esta tarea se debe ejecutar la herramienta Ant con el *target* "add-services", el cual corre un *script* ubicado en el directorio "`sql/`" que agrega finalmente estos tres valores en la tabla "service_types".

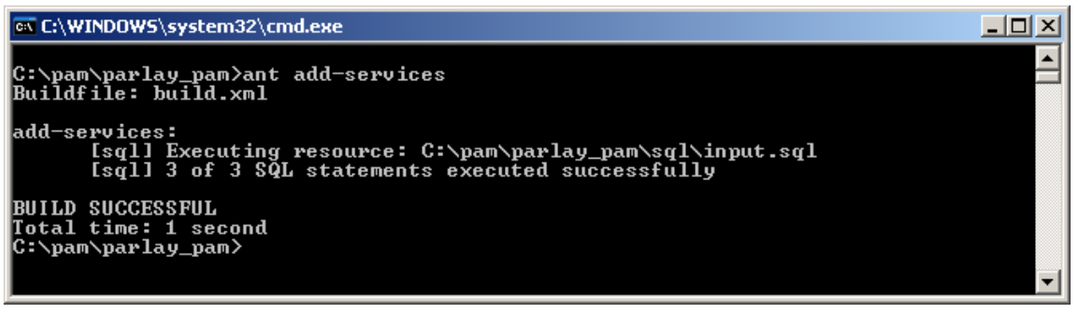


Figura 46. Registro de los SCFs en la base de datos usando la herramienta Ant

5.3.9.3 Creación del entorno de ejecución

Se crearon los siguientes *scripts* (`.bat/.cmd` para Windows, y `bash` para Unix), los que fueron alojados en el directorio "`bin/`" del proyecto PAM:

- `serviceAccess.bat / serviceAccess`: ejecuta el archivo .jar correspondiente al servicio PAM *Access* llamando a su clase principal `ServiceMain.java`
- `serviceProvisioning.bat / serviceProvisioning`: ejecuta el archivo `.jar` correspondiente al servicio PAM *Provisioning* llamando a su clase principal `ServiceMain.java`
- `serviceEvent.bat / serviceEvent`: ejecuta el archivo `.jar` correspondiente al servicio PAM *Event* llamando a su clase principal `ServiceMain.java`
- `environment.cmd`: agrega al sistema las variables de entorno necesarias para poder lanzar los tres SCFs de PAM mediante la ejecución de los tres *scripts* anteriores.

5.3.10 Puesta en marcha del servicio PAM

La puesta en marcha del servicio PAM consiste en ejecutar los *scripts* creados para la lanzar las tres SCFs de PAM (listados en el punto 5.3.9.3). Antes de esto, es requisito que se hayan realizado todos los pasos de preparación para la ejecución del proyecto, descritos en el punto 5.3.9, además de asegurase que la base de datos usada por PAM esté en ejecución, junto con el *framework* y el servidor de nombres (referirse al punto 5.2.5 "Puesta en marcha del *framework* y del servidor de nombres").

Primero, se crean las variables de entornos necesarias mediante la ejecución del *script* `environment.cmd` ubicado en "`bin/`". Luego, en este mismo directorio, se ejecutan los *scripts* que lanzan finalmente las tres SCFs de PAM:

Lanzamiento del servicio PAM *Access*:

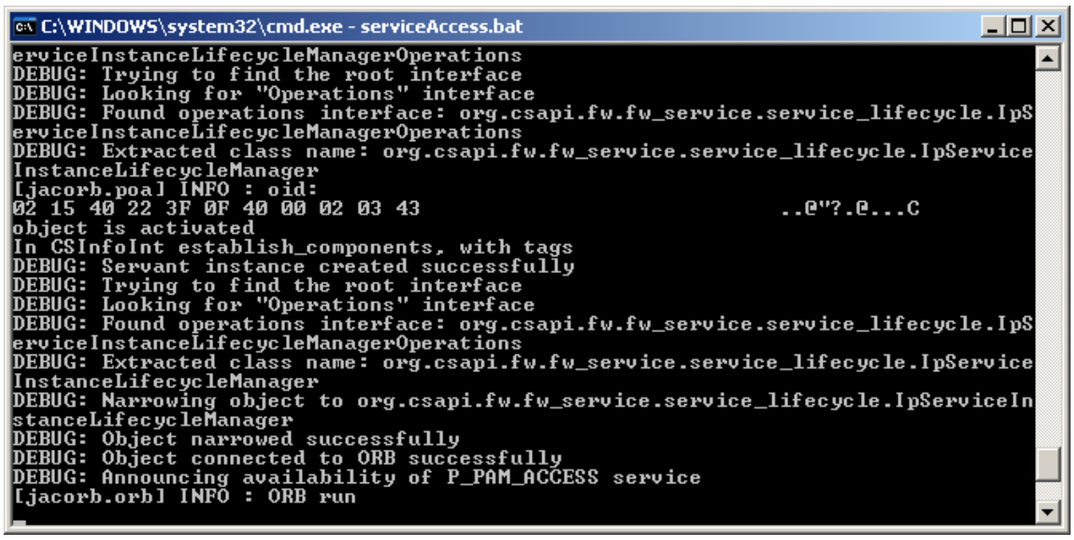


Figura 47. Puesta en marcha de PAM SCF Access mediante la ejecución del script *serviceAccess.bat*

Lanzamiento del servicio PAM *Provisioning*:

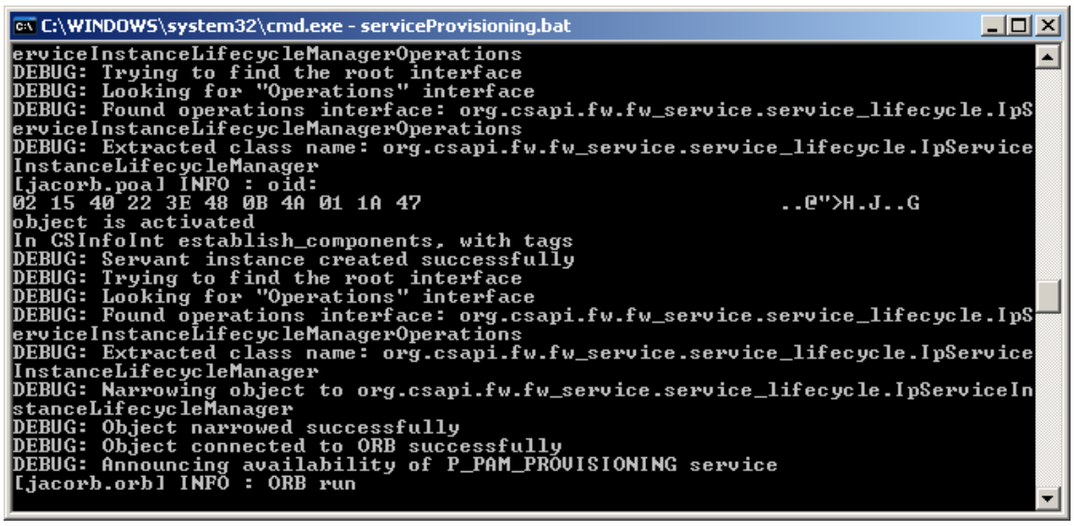


Figura 48. Puesta en marcha de PAM SCF Provisioning mediante la ejecución del script *serviceProvisioning.bat*

Lanzamiento del servicio PAM *Event*:

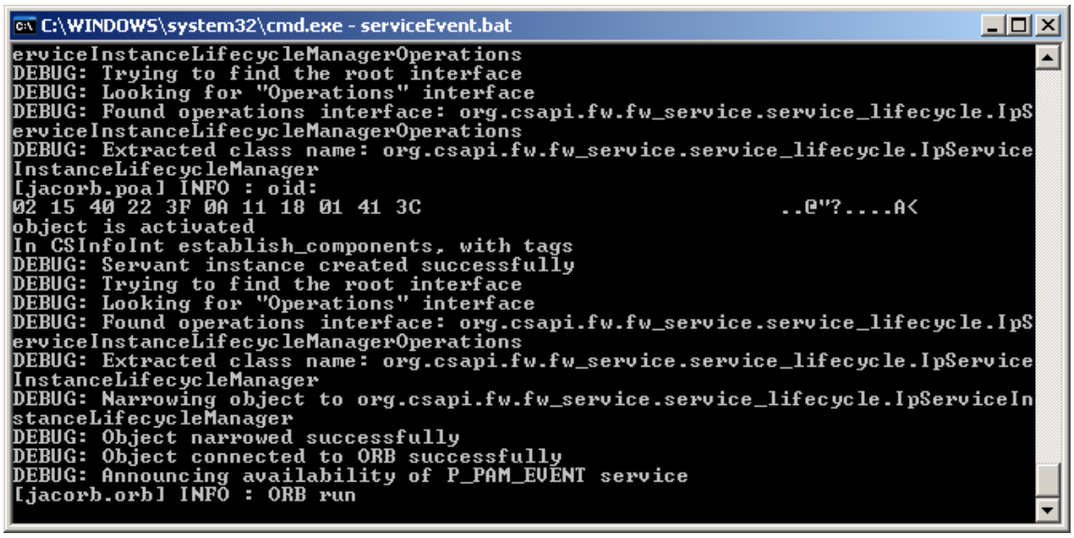


Figura 49. Puesta en marcha de PAM SCF Event mediante la ejecución del script *serviceEvent.bat*

Es importante recordar que el lanzamiento de los tres servicios consiste en la ejecución de sus clases principales respectivas llamadas "`ServiceMain.java`", cuya implementación fue detalladamente descrita en el punto 5.3.8, por consiguiente, todos los mensajes de información (INFO y DEBUG) mostrados en las consolas MS-DOS en las figuras 47, 48 y 49, corresponden a los pasos realizados en la comunicación *Framework*-SCF implementados en "`ServiceMain.java`".

5.4 Implementación de aplicaciones de prueba

Se crearon tres aplicaciones cliente para realizar pruebas de funcionamiento de las interfaces implementadas para las tres SCFs que componen PAM. Estas tres aplicaciones implementan los pasos requeridos para que una aplicación pueda hacer uso finalmente de una capacidad de servicio o SCF (pasos descritos en el punto 3.1.1.4). Estos pasos se listan a continuación, en orden de ejecución, junto con una descripción y las interfaces PAM utilizadas (también se utilizan algunas interfaces del *framework*):

- **Autenticación:** la autenticación de una aplicación cliente en el *framework* sigue los mismos procedimientos que la autenticación de una SCF en el *framework*, junto con utilizar las mismas interfaces y métodos. Este proceso de autenticación fue detalladamente descrito en el punto 5.3.8.

- **Solicitud de la interfaz de descubrimiento:** una vez realizada la autenticación, la aplicación solicita vía CORBA al *framework* una referencia de su interfaz de descubrimiento `IpServiceDiscovery`.

- **Descubrimiento de servicios:** La aplicación, usando la referencia a `IpServiceDiscovery` obtenida, llama al método `listServiceTypes()` de esta interfaz, obteniendo un *array* de *strings* con la lista de SCFs disponibles en la pasarela Parlay/OSA.

- **Selección de servicio + firma de acuerdos de servicio:** luego de obtener la lista de servicios disponibles, la aplicación selecciona la SCF que desea utilizar, y llamando al método `discoverService()` de `IpServiceDiscovery` solicita un identificador único de servicio "`ServiceID`", conocido sólo por el *framework* y por la aplicación.

  Respecto a los acuerdos de servicio, en las aplicaciones creadas se implementó la interfaz cliente `IpAppServiceAgreementManagement`, descrita en el documento de especificación del framework "parte 3" [Ets05c], la cual es usada por el *framework* para solicitar a la aplicación cliente que firme los acuerdos establecidos para el uso del servicio. Este proceso se realiza llamando al método `signServiceAgreement()` de la interfaz implementada. Si la aplicación cliente no tiene permisos para hacer uso de la SCF, una excepción es lanzada.

- **Retorno del administrador del servicio:** Como se indicó en los puntos 5.1.3.3, 5.1.3.4 y 5.1.3.5 en la descripción de las interfaces de cada SCF de PAM, existe una "Interfaz Administrador" para cada SCF: `IpPAMProvisioningManager`, `IpPAMPresenceAvailabilityManager` e `IpPAMEventManager`.

  La llamada al método `signServiceAgreement()` realizada en el paso anterior, retorna una referencia a la Interfaz Administrador de la SCF que se solicita.

- **Uso del servicio:** Utilizando la referencia a la interfaz administrador del servicio, vía CORBA, se pueden obtener las referencias a las interfaces restantes de la SCF en uso, y realizar invocaciones remotas de sus métodos implementados.

5.4.1 Prueba del la SCF PAM Event.

Para realizar pruebas de funcionamiento de las interfaces implementadas en la SCF PAM *Event*, utilizada para el manejo de notificaciones de eventos vía *callbacks*, se crearon dos aplicaciones:

1. "*Consumer*": levanta una interfaz grafica en donde, mediante un esquema de árbol, se muestran las identidades, agentes, y los atributos que se van creando en el servidor en tiempo de ejecución. Esta aplicación utiliza la interfaz cliente `IpAppPAMEventHandle`, descrita en el punto 5.1.3.5, la que fue implementada para que "*Consumer*" reaccione a los eventos ocurridos sobre agentes e identidades, y sus atributos, los cuales son notificados a la aplicación desde la pasarela Parlay/OSA, específicamente desde la SCF PAM *Event*.

2. "*Application*": Esta aplicación ejecuta los métodos necesarios para crear un conjunto de atributos y tipos, algunas identidades y agentes de ejemplo, mientras "*Consumer*" reacciona a los eventos generados por las acciones efectuadas por "*Application*". En esta aplicación también se puso a prueba en cierta forma las SCFs PAM *Provisioning* y PAM *Access*, utilizadas para crear identidades, agentes y sus atributos.

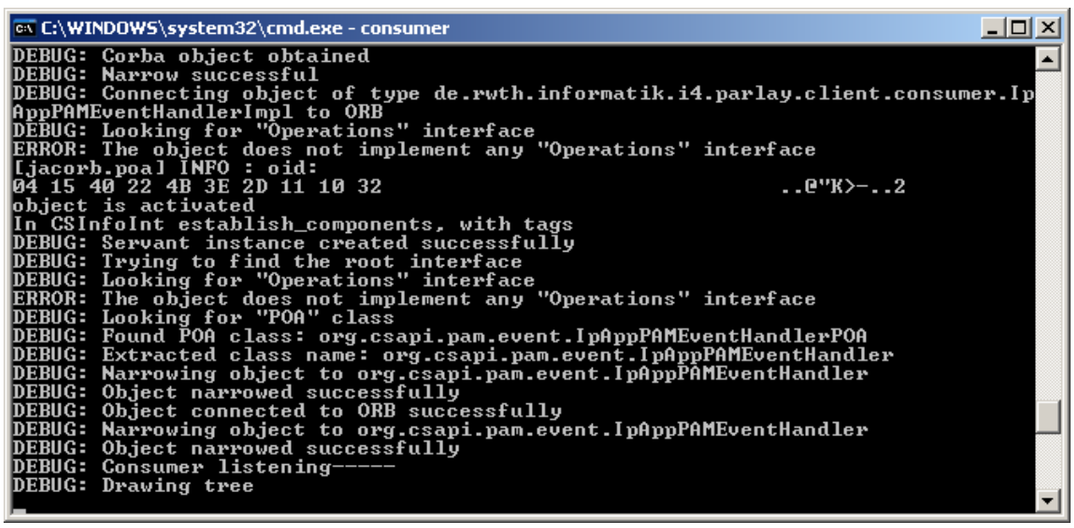


Figura 50. La aplicación "consumer" en ejecución, quedando en espera a la notificación de eventos desde la pasarela de Parlay/OSA.

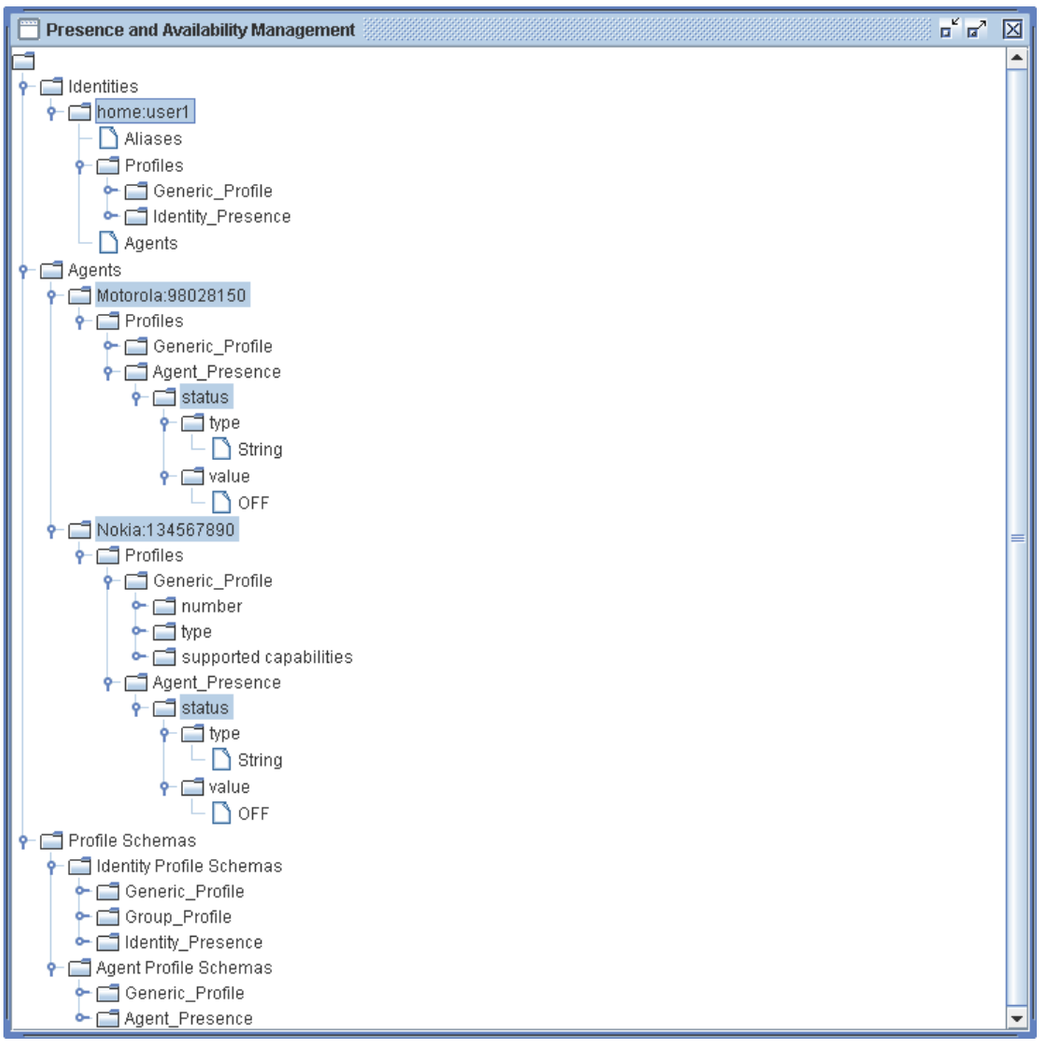


Figura 51. Aplicación cliente "*Consumer*": estructura de árbol que reacciona a los eventos lanzados al crear agentes, identidades y atributos.

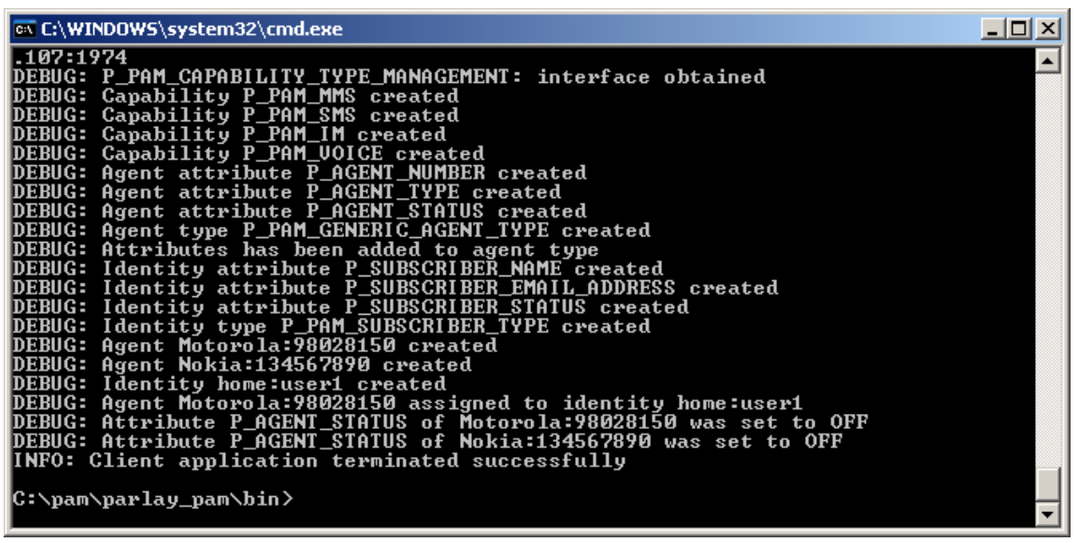


Figura 52. Creación de identidades, agentes y algunos atributos de ejemplos ejecutada por la aplicación cliente "*Application*".

5.4.2 Prueba del las SCFs PAM Provisioning y PAM Access

Para poner a prueba las interfaces implementadas en PAM *Provisioning* y PAM *Access*, se creó una aplicación llamada “Gui”, que levanta una interfaz gráfica a través de la cual se probaron las siguientes funcionalidades:

Para PAM Provisioning

Sección “*Identity Management*”:

- Crear/eliminar Identidad o grupos de identidades.
- Administrar miembros de un grupo
- Administrar aliases
- Crear y asociar atributos de identidad

Sección “*Agent Management*”:

- Crear/eliminar agentes
- Administrar capacidades
- Crear y asociar atributos de agente

Sección “*Agent Assignment Management*”:

- Asociar agentes a identidades

Para PAM Access

Sección “*Identity presence and availability*”:

- Crear y asociar atributo de presencia y disponibilidad de identidades

Sección “*Agent presence*”:

- Crear y asociar atributo de presencia de agentes

En las Figuras 53, 54, 55 y 56 se muestran imágenes del la aplicación “Gui” implementada:

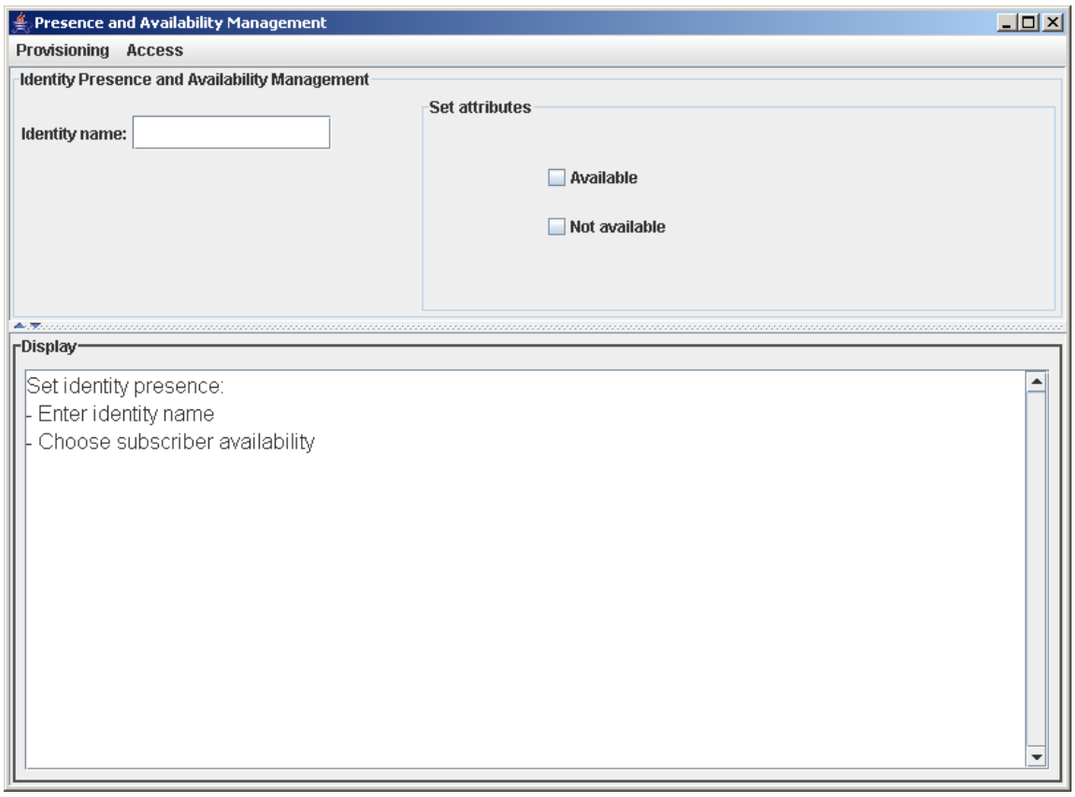


Figura 53. Aplicación cliente “Gui”: ingreso de atributos de presencia para una identidad

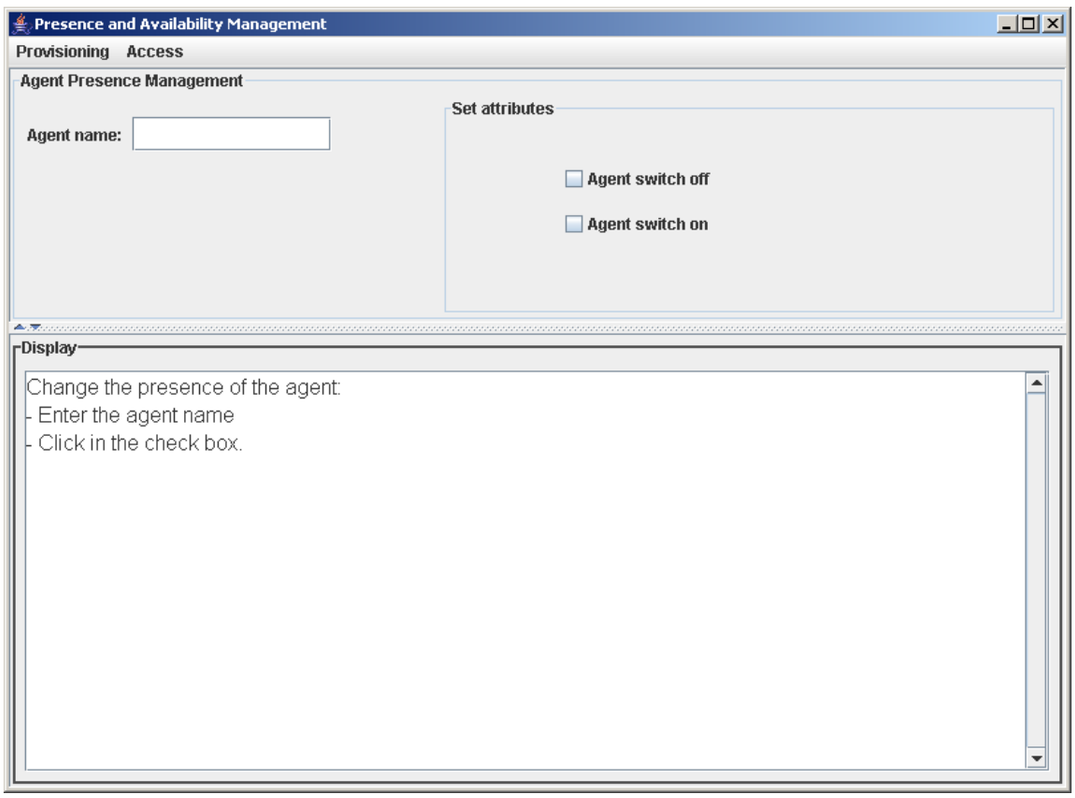


Figura 54. Aplicación cliente “Gui”: ingreso de atributos de presencia para un agente

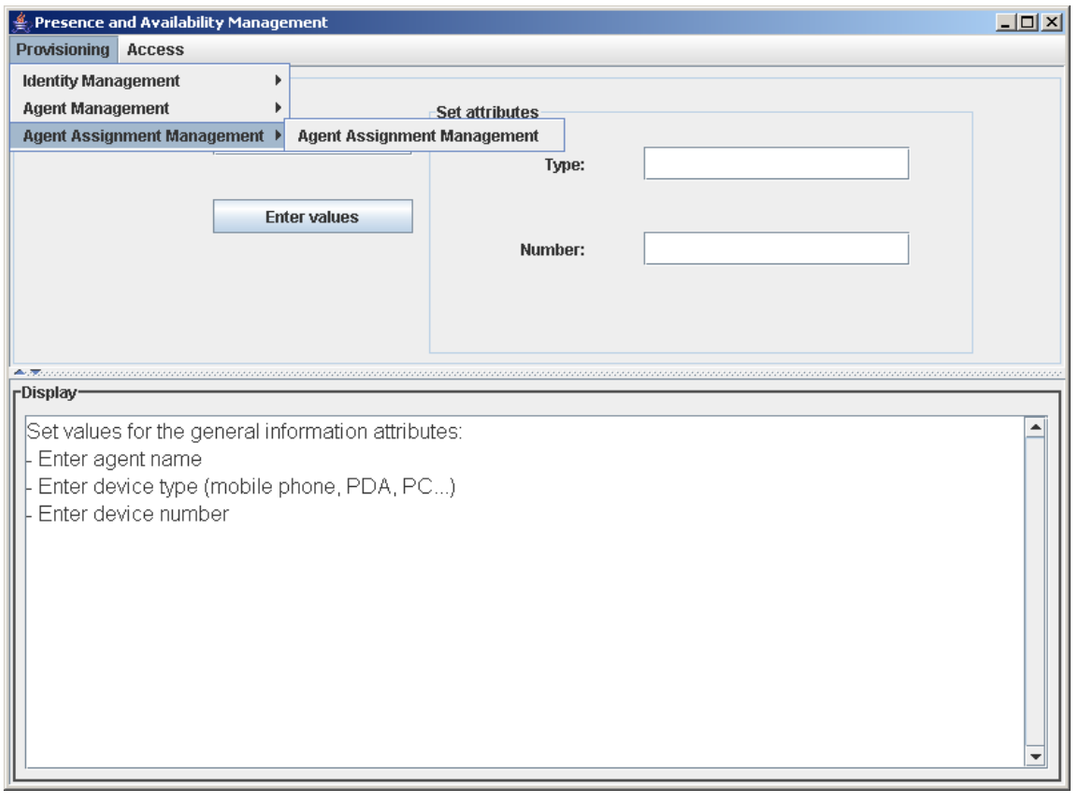


Figura 55. Aplicación cliente "Gui": opciones disponibles en el menú "Provisioning".

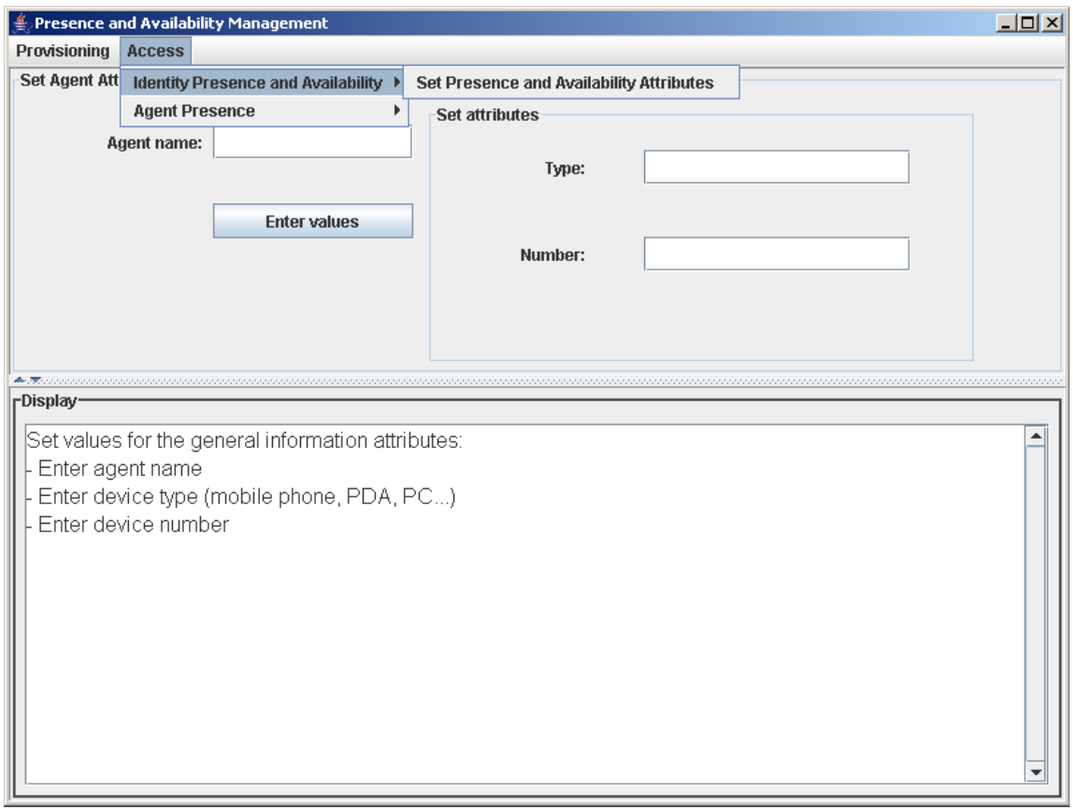


Figura 56. Aplicación cliente "Gui": opciones disponibles en el menú "Access".

CAPÍTULO VI: APLICACIÓN PARLAY-X: LOCALIZACIÓN DE USUARIOS A TRAVÉS DE GOOGLE MAPS

6.1 Descripción de la aplicación

Con el propósito de mostrar de una forma más práctica el potencial de las APIs de Parlay-X, se implementó una aplicación que hace uso del Servicio Web "*Terminal Location*" especificado en Parlay-X 2.1, por medio del cual se obtiene información de localización de un terminal determinado, la que es usada para mostrar a través de un mapa en una página Web, la ubicación de un usuario utilizando la API de Google Maps.

6.2 "Terminal Location": Servicio Web de Parlay-X

El Servicio Web "*Terminal Location*" se encuentra implementado en el emulador para servicios de telecomunicaciones que Ericsson pone a disposición de los desarrolladores para la creación de aplicaciones Parlay-X. Las características de este emulador y el servicio de localización de terminales, fueron debidamente descritos en el Capitulo IV. En la implementación de la aplicación Parlay-X, se utilizó este emulador para obtener información de localización de un terminal, a través del Servicio Web "*Terminal Location*".

6.2.1 Interfaz gráfica del servicio de localización del emulador

Retomando lo descrito en el punto 4.3.4 referente al servicio de localización de terminales del emulador, la emulación de la localización de un terminal determinado se realiza a través una interfaz gráfica en donde se despliega un mapa y los terminales creados en el emulador dispuestos sobre este mapa de acuerdo las coordenadas (latitud y longitud) de ubicación de éstos. El mapa originalmente integrado en el emulador corresponde a una zona de Estocolmo, Suecia (mostrado en la figura 29 en el Capítulo IV). Este mapa fue reemplazado por uno que muestra una zona de la ciudad de Valdivia, en donde las coordenadas límites correspondientes a la longitud izquierda, longitud derecha, latitud superior y latitud inferior, fueron ingresadas en la sección de configuración del servicio de localización del emulador (descrita en el punto 4.3.4).

La interfaz grafica del emulador, sección de localización de terminales, con el nuevo mapa se muestra en la figura 57:

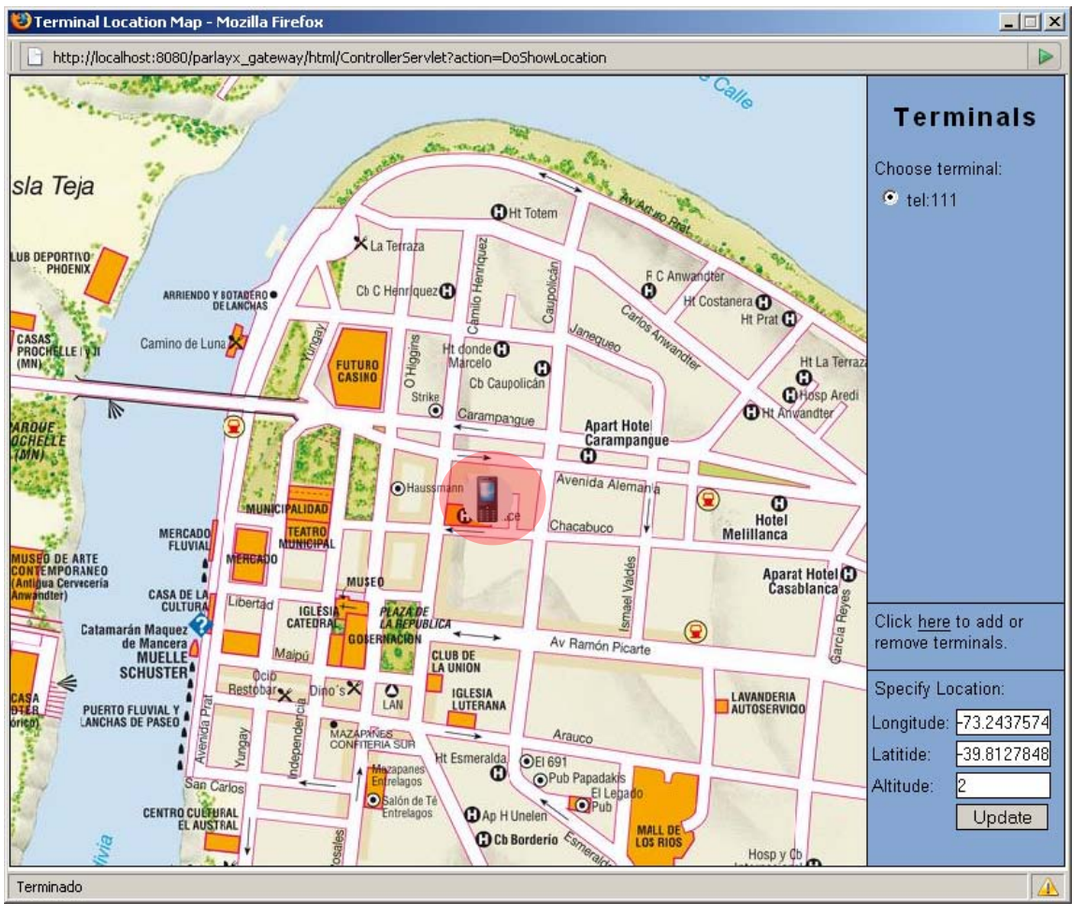


Figura 57. Interfaz gráfica del servicio de localización del emulador mostrando un mapa de una zona de la ciudad de Valdivia.

Las coordenadas de ubicación de un terminal pueden ser especificadas arrastrando dentro del mapa el icono que representa al móvil (marcado en verde en la figura anterior), o indicando directamente la longitud/latitud en el formulario ubicado en la parte inferior derecha.

La aplicación Parlay-X a implementar conecta con el Servicio Web "*Terminal Location*" del emulador, para obtener la latitud/longitud de un terminal móvil determinado, especificada en la interfaz grafica del servicio de localización del emulador mostrada en la figura 57. Esta información de localización es utilizada por la aplicación Parlay-X para desplegar un mapa con la ubicación del terminal a través de una página Web utilizando Google Maps.

6.3 API de Google Maps

Google Maps es un servicio gratuito que ofrece tecnología cartográfica potente y fácil de usar. La API de Google Maps permite la incrustación de mapas en cualquier página Web mediante JavaScript, proporcionando varias utilidades para la manipulación de éstos.

Para cargar la API en una página Web, sólo se debe incluir la siguiente línea de código JavaScript:

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abcdefg"
type="text/javascript"></script>
```

La URL indicada dirige a la ubicación del archivo JavaScript que incluye todos los métodos utilizados para manejar un mapa.

Para mostrar el mapa en una página Web, se debe reservar un lugar para él mediante la creación de un elemento `div` indicando el ancho y largo deseable. La carga del mapa se realiza creando un objeto `GMap2` e indicando como parámetro el elemento `div` en donde éste será desplegado. Luego, se debe indicar la coordenada inicial (latitud, longitud) correspondiente al centro del mapa, a través de la invocación al método `setCenter()`, señalando también el nivel de acercamiento (*zoom*).

```
var map = new
GMap2(document.getElementById("div_creado_para_el_mapa"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
```

Los pasos anteriormente señalados son los mínimos que se deben realizar para desplegar un mapa. La API de Google Maps entrega muchos otros métodos para trabajar sobre los mapas cargados. En la implementación de la aplicación Parlay-X se utilizaron funcionalidades de la API para crear y situar una imagen sobre el mapa que represente al terminal móvil, además de funcionalidades para ir moviendo el mapa de forma dinámica y automática según cambie la ubicación del terminal.

6.4 Implementación de la aplicación Parlay-X

Para realizar la implementación de la aplicación Web de localización de usuarios, se utilizó el entorno de desarrollo Eclipse Ganymede 3.4.0 (versión para desarrolladores Java EE) en donde se creó un nuevo Proyecto Web Dinámico denominado "`parlay_user_location`" (ver Fig.58).

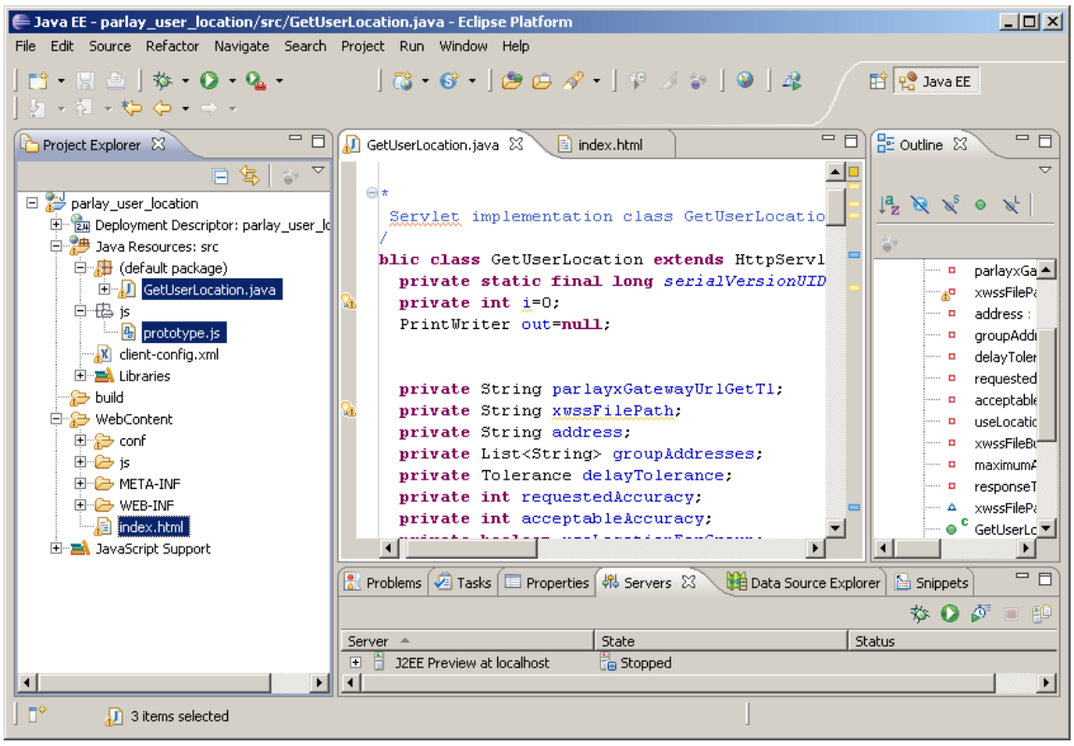


Figura 58. Entorno de desarrollo Eclipse usado para implementar la aplicación Parlay-X.

El acceso al Servicio Web "*Terminal Location*" implementado en el emulador, se efectuó utilizando los componentes Java SE incorporados en el SDK de Ericsson (ver punto 4.1.2), específicamente los JavaBeans `GetTerminalLocationBean.class` y `LocationInfoBean.class`. Para implementar la lógica necesaria al momento de consultar por la localización de un terminal determinado, se creó un *servlet* llamado `GetUserLocation` el cual hace uso de los JavaBeans indicados anteriormente para acceder al Servicio Web "*Terminal Location*" y obtener la información de localización. Al momento de usar los Componentes Java SE, para realizar una comunicación segura con los Servicios Web del emulador, se requiere del uso de un archivo de seguridad XWSS[20] definido a través del archivo `client-config.xml`, en donde se especifica un "`username token`" y un "`password`", cuyos valores deben coincidir con los especificados en la configuración del emulador.

La invocación al servlet `GetUserLocation` se realiza desde una página HTML (`index.hml`) creada para mostrar finalmente un mapa de Google Maps con la ubicación actual de un terminal determinado, utilizando la información de localización

[20] XML Web Services Security

entregada al llamar al *servlet*, retornando la longitud y latitud actual del móvil. Las coordenadas de ubicación del terminal son actualizadas invocando a `GetUserLocation` cada un segundo de forma asíncrona utilizando AJAX[21], específicamente haciendo uso de la librería *Prototype*[22]. La actualización del mapa de acuerdo a la ubicación actual del móvil también se realiza mediante AJAX.

La figura 59 representa de forma gráfica los componentes y tecnologías involucradas en la implementación de la aplicación Parlay-X, indicando también las interacciones entre estos componentes.

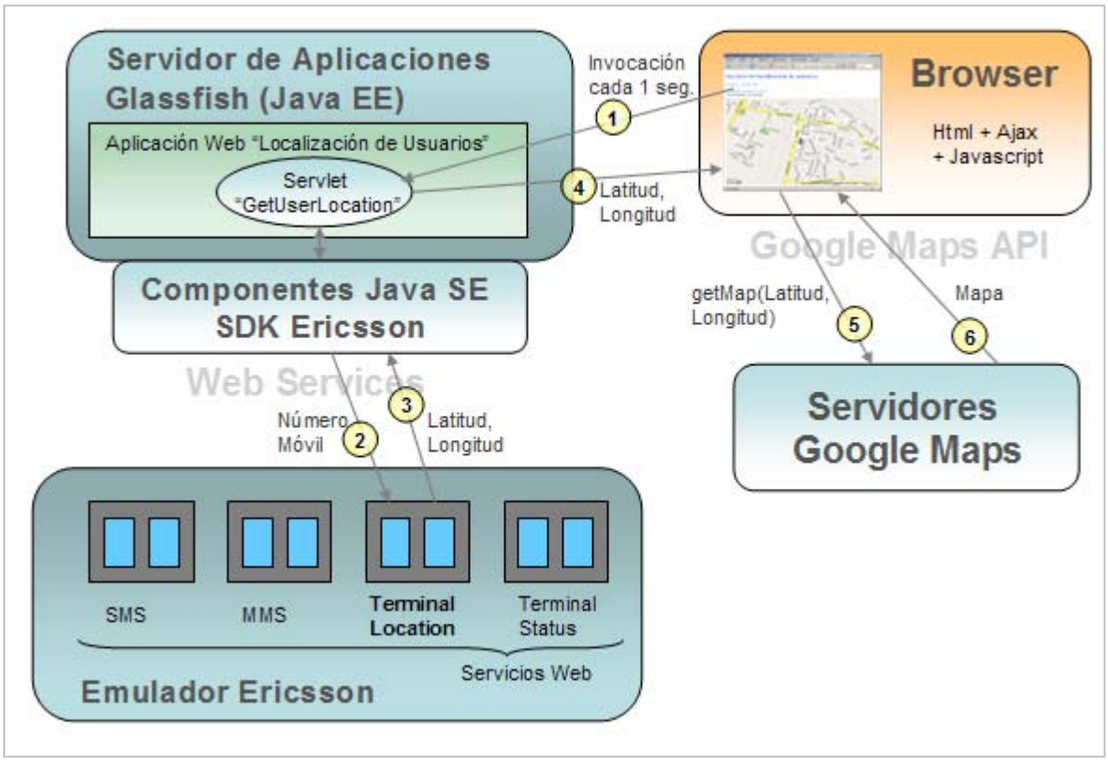


Figura 59. Componentes y tecnologías empleadas en la implementación de la aplicación, junto con la interacción entre los componentes.

A continuación se describirá, en base a la figura anterior, cada paso de interacción realizado entre cada componente. El ciclo de pasos se repite cada un segundo.

1. Desde `index.hml` se invoca al servlet `GetUserLocation` para iniciar la obtención de la información de localización de un terminal determinado.

[21] AJAX: "Asynchronous JavaScript And XML"
[22] Framework JavaScript orientado al desarrollo rápido y sencillo de aplicaciones Web basadas en AJAX.

2. El *servlet* `GetUserLocation` realiza una llamada al Servicio Web "*Terminal Location*" haciendo uso de los componentes Java SE, e indicando el número del terminal móvil que se desea localizar.
3. El Servicio Web "*Terminal Location*" retorna al *servlet* la longitud y latitud actual del móvil consultado.
4. La información de localización obtenida es entregada de forma asíncrona al *Browser* que desplegará el mapa junto con el terminal y su ubicación dentro de éste.
5. Utilizando la longitud y latitud obtenida, y a través de la API de Google Maps, se invoca al método `getMap()`para solicitar el mapa centrado en las coordenadas de localización indicadas.
6. El mapa solicitado es retornado y desplegado en el *Browser*.

6.5 Ejecución de la aplicación Parlay-X

Una vez realizada la implementación de la aplicación, el proyecto se exportó a un archivo `.war` con nombre `parlay_user_location.war`, equivalente a un archivo `.jar`, pero creado para ser ejecutado en un servidor de aplicaciones. La aplicación Web implementada fue instalada en el servidor de aplicaciones *Java Glassfish* a través de la consola de administración, cargando el archivo `.war` correspondiente.

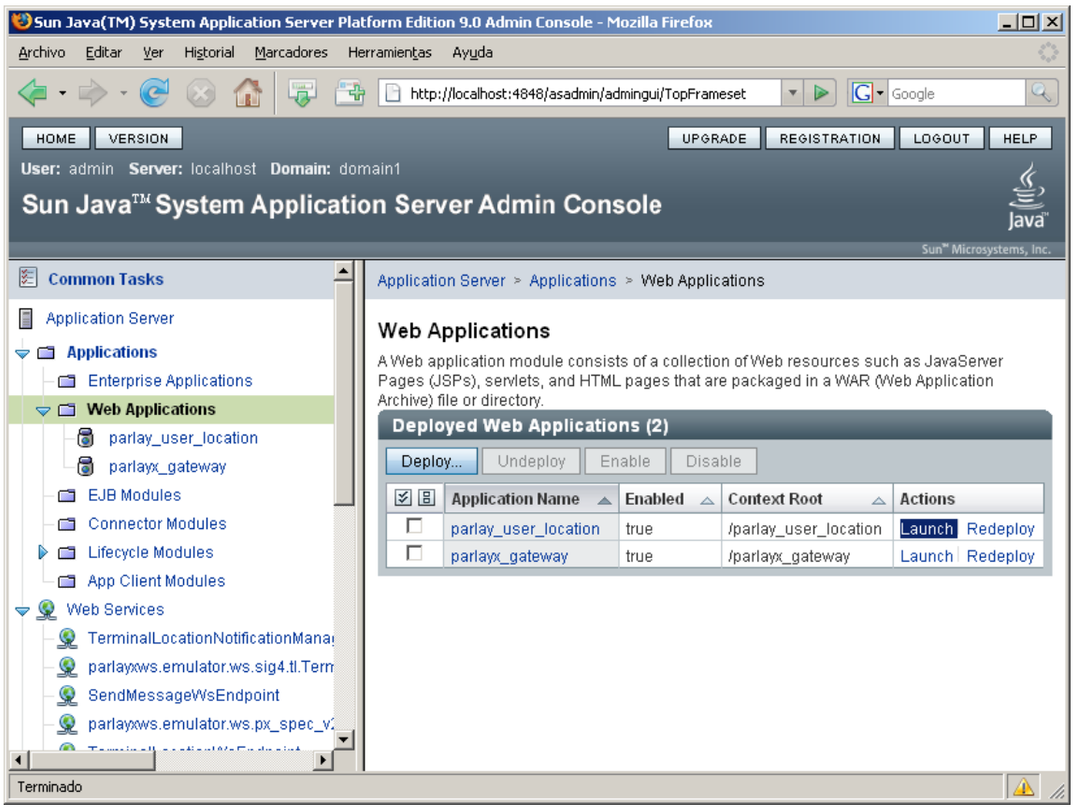


Figura 60. Carga de la aplicación Web creada en el Servidor de Aplicaciones Java Glassfish.

Una vez cargada la aplicación Web en Java Glassfish, para proceder con su ejecución, se presiona en la opción "*Launch*" (marcada en azul en la figura 60), la cual lanza la aplicación en una nueva ventana en el *Browser*, a través de la siguiente URL:

```
http://localhost:8080/parlay_user_location/
```

Es requisito tener en ejecución el emulador de Ericsson junto con un terminal móvil creado.

Para poner a prueba la aplicación Parlay-X, se creó en el emulador un terminal móvil identificado con el número "111".

A medida que se ingresa una nueva ubicación para el móvil "111" dentro los rangos definidos por el mapa cargado, ya sea arrastrando la imagen del terminal sobre el mapa, o ingresando directamente una longitud y latitud (siempre dentro de los límites del mapa cargado), la aplicación Web de localización de usuarios implementada actualiza cada un segundo, a través de Google Maps, la nueva ubicación del terminal representado con una imagen de un teléfono móvil.

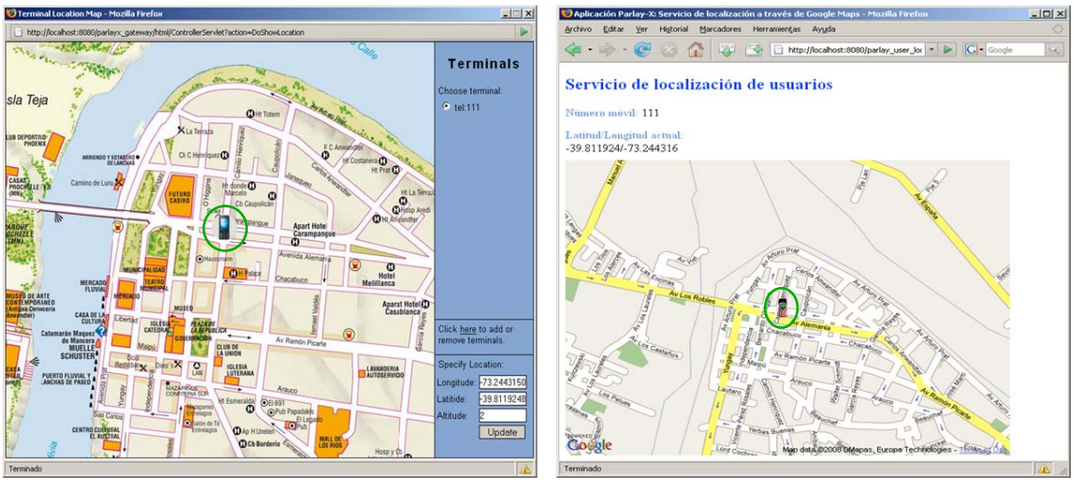


Figura 61. Fig. izquierda, el emulador y su interfaz gráfica de localización de terminales; fig. derecha, aplicación Parlay-X para la localización de usuarios.

En la figura 62, y observando la figura 61, se puede apreciar la actualización de la ubicación del terminal en la aplicación Parlay-X, realizada a través del desplazamiento del móvil en el emulador.

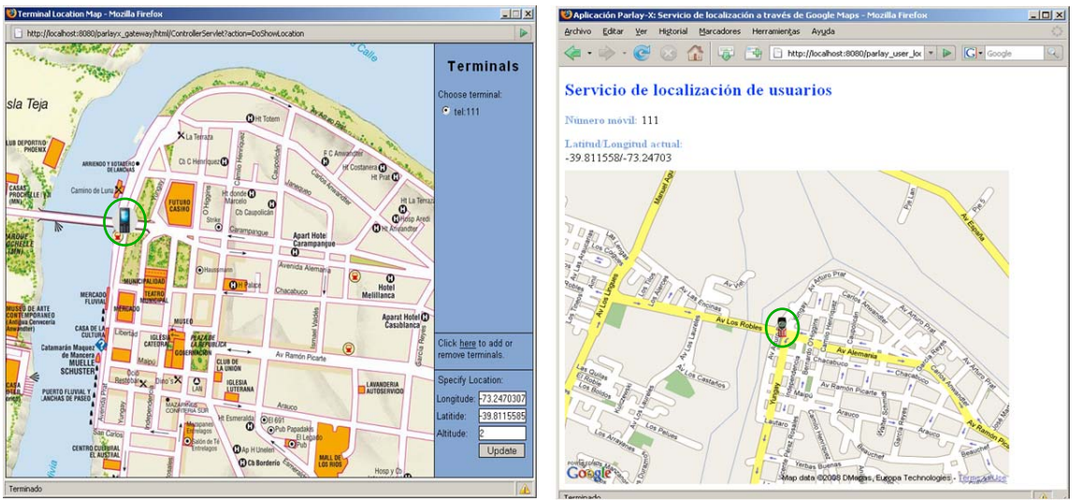


Figura 62. Fig. izquierda, el emulador y la nueva ubicación del terminal; fig. derecha, aplicación Parlay-X desplegando la nueva ubicación.

La aplicación implementada, muestra también información actualizada de la ubicación dada por la latitud/longitud actual del terminal.

CAPÍTULO VII: SITUACION ACTUAL DE INSERCIÓN DEL ESTANDAR A NIVEL NACIONAL

7.1 Encuesta realizada y resultados

La siguiente encuesta fue enviada a las principales compañías nacionales de telecomunicaciones con el objetivo principal de analizar el nivel de inserción de Parlay/OSA y Parlay-X en Chile:

1. *¿Qué nivel de conocimiento tenía sobre el estándar Parlay/OSA y Parlay-X?*
   - ☐ alto
   - ☐ intermedio
   - ☐ básico
   - ☐ nulo

2. *Respecto de la incorporación de este estándar, actualmente la empresa:*
   - ☐ lo tiene ya incorporado como plataforma de servicios.
   - ☐ aún no lo incorpora, y no existen planes para hacerlo.
   - ☐ aún no lo incorpora, pero sí está planificado hacerlo.
   - ☐ aún no lo incorpora, y no se sabe si lo hará.
   - ☐ se encuentra en fase de incorporación.

3. *En caso que el estándar no se encuentre incluido como plataforma de servicios en la empresa, a simple vista ¿cómo ve la oportunidad de incorporarlo?*
   - ☐ muy atractiva
   - ☐ atractiva
   - ☐ poco atractiva. Justifique:
   - ☐ nula. Justifique:

4. *En caso que el estándar ya se encuentre integrado en la empresa, su incorporación ha resultado:*
   - ☐ muy beneficiosa
   - ☐ beneficiosa
   - ☐ poco beneficiosa. Justifique:

Se obtuvieron las siguientes respuestas:

Empresa: ENTEL PCS

Contacto: Sr. Claudio Cerda Salvo (Subgerente Ingeniería Desarrollo de Servicios)

Repuestas:

1. intermedio
2. aún no lo incorpora, y no existen planes para hacerlo.
3. nula. Justifique: Existen alternativas costo – eficientes mejores

Empresa: VTR

No se pudieron obtener las respuestas

Empresa: MOVISTAR

No se pudieron obtener las respuestas

7.2 Análisis

Después de una espera prudente de tiempo, finalmente no se pudo obtener la cantidad de respuestas requeridas y necesarias para lograr un análisis acabado mediante la encuesta realizada. En vista del poco *feedback* logrado, se realizaron contactos telefónicos para poder obtener más información al respecto. Las personas contactadas indicaron no tener dominio del tema, señalando además que en sus respectivas empresas se emplean soluciones propietarias para el desarrollo de servicios de telecomunicaciones.

También fue posible obtener información de la empresa "Telefónica del Sur" en donde Parlay tampoco se encuentra integrado como plataforma de servicios, en su lugar se utilizan también plataformas propietarias.

Uno de los grandes potenciales de Parlay es la posibilidad de abrir la red a terceros, está característica en Chile aún no se está empleando mediante la incorporación de APIs estándares, en su lugar, se utilizan soluciones propietarias desarrolladas particularmente para cada funcionalidad de red principalmente en el ámbito de la telefonía móvil.

CONCLUSIONES

Por estos días, se habla mucho de las llamadas Redes de Próxima Generación, término empleado para referirse a una red global "*all-IP*" que unifica todas las redes de telecomunicaciones, independiente de la forma de acceso (fija, inalámbrica, móvil), y en donde Parlay/OSA junto a Parlay-X juegan un rol fundamental en el plano se Servicios/Aplicaciones de la arquitectura de estas nuevas redes, la que también es integrada por importantes tecnologías habilitantes para la convergencia como lo son IMS y el protocolo SIP.

Parlay/OSA con sus 10 años de madurez ha ido formando parte en la solución de despliegue de servicios en muchas compañías de telecomunicaciones, dando la posibilidad a terceras partes hacer uso de las capacidades de las redes para desarrollar aplicaciones de negocio de forma ágil para una rápida puesta en marcha, a través de unas APIs bien definidas, estandarizadas, libres, seguras, abiertas y agnósticas tecnológicamente, siendo éstas un punto de unión en la convergencia de dos mundos que en un principio nacieron de forma separada: las tecnologías de la información y las telecomunicaciones. Por otro lado, Parlay-X nació como una iniciativa para simplificar el modelo de Parlay/OSA cuyas APIs son implementadas principalmente utilizando tecnología CORBA, y se crearon con el objetivo de llegar a un mayor número de desarrolladores de aplicaciones, puesto que con Parlay-X las funcionalidades de las redes de telecomunicaciones quedan abiertas a través de Servicios Web, tecnología que ha ido tomando bastante fuerza y protagonismo en estos últimos años.

El servicio PAM o "Manejo de Disponibilidad y Presencia" seleccionado para ser implementado en la presente tesis, se encuentra definido y diseñado bajo UML de acuerdo a las especificaciones de Parlay/OSA versión 5.0. Este servicio establece un estándar para el mantenimiento, recuperación y publicación de información de presencia y disponibilidad. PAM fue implementado bajo CORBA y Java (requisitos del laboratorio) e integrado con un *framework* Parlay/OSA el cual fue facilitado con el fin de poder gestionar el servicio PAM para que las aplicaciones que deseen hacer uso de este servicio puedan acceder a él de forma controlada y segura. Se crearon tres aplicaciones cliente conectadas con la pasarela Parlay/OSA (compuesta por el *framework* y por PAM) con las cuales se probó el correcto funcionamiento de las interfaces implementadas del servicio.

Para dar conocer de forma práctica el potencial de Parlay-X se implementó una sencilla aplicación Web de “Localización de Usuarios” resultando esta implementación menos compleja y más rápida que la implementación de las aplicaciones Parlay/OSA bajo CORBA, en contraste con la utilización de Servicios Web en la aplicación Parlay-X.

Con el desarrollo de la presente tesis se obtuvo un gran *know-how* en cuanto a la comprensión de documentos técnicos de especificación de estándares como los que definen a las APIs de Parlay/OSA, llegando a conocer y comprender muy por dentro las APIs al haber implementado un servicio basado en este estándar. Además, se identificaron los beneficios que trae consigo la incorporación de Parlay/OSA y Parlay-X como plataforma de servicios en las compañías de telecomunicaciones, por medio de la integración de infraestructura *legacy* con nuevas plataformas para crear servicios de valor agregado.

La adopción de este estándar por parte de grandes compañías de telecomunicaciones en el mundo, principalmente europeas, además de operadores en Brasil, Colombia y México, habla de la aceptación real y beneficiosa que ha tenido Parlay. En base a la encuesta y los contactos telefónicos realizados para obtener información del nivel de penetración de esta tecnología a nivel nacional, se concluye que en Chile no se está trabajando con esta tecnología, o al menos no se encontraron evidencias que puedan ser presentadas en este trabajo de tesis luego de haber consultado a diversas empresas de telecomunicaciones a nivel nacional, debido principalmente a que no se conoce mucho sobre este estándar dado que las tecnologías empleadas en las soluciones de servicios se basan en soluciones propietarias y particulares. Se espera que este trabajo de titulo sea un referente de Parlay en Chile.

REFERENCIAS

[Aba02] Abarca C., Bennett A., Moerdijk A. & Unmehopa M. (2002). Meeting #16 of Parlay, ETSI Project OSA, 3GPP TSG_CN WG5 - Parlay/OSA: an open API for service development. (Hong Kong, China, 8 February 2002).

[Ber06] Bermúdez C. (2006). Conferencia: La convergencia en redes móviles vista por los operadores, en IV Foro Iberoamericano “AHCIET MÓVIL”. (San José, Costa Rica, Octubre 2006).

[Cit06] CITEL - Comisión Interamericana de Telecomunicaciones (2006), Redes de próxima generación - Visión general de normas. Carpeta Técnica-1 rev.-5. Septiembre.

[Day00] Day M., Rosenberg J. & Sugano H. (2000). RFC 2778: A Model for Presence and Instant Messaging, IETF – Network Working Group. Frebruary. Disponible en http://www.ietf.org/rfc/rfc2778.txt, consultado en Julio 2008.

[Det07] Detecon Consulting (2007). Business Trends & Challenges towards full-IP, en C5 World Forum: Customer-Centric Converged Communication and Content. (Milano, Italy, 28 March 2007).

[Eri06] Ericsson (2006), Efficient softswitching White Paper, 284 23-8107 Uen Rev. A, August.

[Eri05] Ericsson (2005), Evolution towards converged services and networks - White Paper, 284 23-3017 Uen Rev B, April.

[Eri07] Ericsson (2007), Introduction to IMS - White Paper, 284 23-8123 Uen Rev A, March.

[Ets06] ETSI (2006). ETSI OSA Parlay X - Parlay X 2.1 Specifications ES 202 391 V1.2.1. December.

[Ets05a] ETSI (2005). Open Service Access (OSA) API – Part1: Overview (Parlay 5) - ES 203 915-1 V1.1.1 (2005-04).

[Ets05b] ETSI (2005). Open Service Access (OSA) API – Part14: Presence and Availability Management SCF (Parlay 5) - ES 203 915-14 V1.1.1 (2005-04).

[Ets05c] ETSI (2005). Open Service Access (OSA) API – Part3: Framework (Parlay 5) - ETSI ES 203 915-3 V1.1.1 (2005-04).

[Ets05d] ETSI (2005). Open Service Access (OSA) API – Part2: Common Data Definitions (Parlay 5) - ETSI ES 203 915-2 V1.1.1 (2005-04).

[Fal04] Falcarin P. & Licciardi C. (2004). Analysis of NGN service creation technologies. Paper based on Eurescom Project P1109 Next Generation Networks.

[Fra07] Fratini C., (2007) Redes de Próxima Generación, Tesis de postgrado EGT. Disponible en http://www.itba.edu.ar/capis/epg-tesis-y-tf/fratini-tfe.pdf, consultado en Octubre de 2007.

[Fre] Frecuencia Online, Portal de los operadores móviles Latinoamericanos. Disponible en http://www.espanol.frecuenciaonline.com/home/contenidos.php?id=41&identificaArticulo=820, consultado en Octubre de 2007.

[Gar06] García M. (2006), Servicios de futuro y convergentes. Estado actual y estrategia de negocio. ETSI Telecomunicación (Universidad Politécnica de Madrid, España, Diciembre del 2006).

[Gar08] Garibay J. & Lewis A. (2008), ¿Es la convergencia una verdadera revolución tecnológica que transformará a la economía global?, Regulatel, (4)11: 1-3. Enero/Marzo.

[Hua08] Huawei (2008). Network Evolution – New Business Model, Samena. (Dubai ,United Arab Emirates, April 2008).

[Ibm06] IBM (2006), Introduction to IP Multimedia Subsystem (IMS), Part 1: SOA Parlay X Web services. Disponible en: http://www.ibm.com/developerworks/webservices/library/ws-soa-ipmultisub1/ consultado en Octubre de 2007.

[Itu04] ITU (2004), Serie Y: Infraestructura mundial de la información, aspectos del protocolo internet y redes de la próxima generación. Redes de la próxima generación – Marcos y modelos arquitecturales funcionales, Recomendación UIT-T Y.2001. Diciembre.

[Itu07] ITU (2007). Telecom/ICT Market Trends, en 8th Forum on Telecommunications/ICT Regulation in Africa (Nairobi-Kenya, Africa, 6-7 June 2007).

[Moe02] Moerdijk A. & Klostermann L. (2002). Opening the networks with Parlay/OSA APIs: Standards and aspects behind the APIs. Draft version, to be resubmitted to IEEE Communications Magazine.

[Omg] OMG. CORBA® BASICS. Disponible en http://www.omg.org/gettingstarted/corbafaq.htm#WhatIsIt, Consultado en Octubre de 2007.

[Par04a] Parlay Group (2004). IMS and Parlay: Finding the optimum strategy for real-world deployments. November.

[Par04b] Parlay Group (2004). Parlay/OSA Overview. May.

[Par05a] Parlay Group (2005). Parlay Business Values: Parlay/OSA - A Key Element of Modern Service Delivery Platforms. January.

[Par05b] Parlay Group (2005). Comparing OMA OSE and Parlay Architectures. March.

[Para] Parlay Group, Productos basados en estandar Parlay/OSA. Disponible en http://wiki.parlay.org/wikipub/index.php?title=Products, consultado en Octubre de 2007.

[Parb] Parlay Group, Investigaciones realizadas sobre el estándar Parlay/OSA. Disponible en http://wiki.parlay.org/wikipub/index.php?title=Parlay_Research, consultado en Octubre de 2007.

[Rau02] Rautela D., Markwardt G. & Kamel A. (2002), Combining Open APIs(Parlay/JAIN) & Software Agents for Next Generation Mobile Services , en First International Workshop on M-Services - Concepts, Approaches, and Tools (Lyon, France, June 26, 2002).

[Sub07] SUBTEL (2007). Servicio de telefonía móvil - Indicadores de penetración de mercado. Diciembre.

[Unm06] Unmehopa M., Vemuri K. & Bennett A. (2006). Parlay/OSA, From Standars to Reality. Wiley.

[Wil06] Wiles A. & Boswarthick D. (2006). Global Standars, the key enabler for the Next Generation Networks, en ICT-OSA/Parlay Workshop (Brazil, March 2006).