Capítulo II

Introducción Teórica

1. Los servicios 3G

1.1 Introducción

Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz y datos (una llamada telefónica) como así como exclusivamente datos (como la descarga de programas, intercambio de correo electrónico y mensajería instantánea).

La tercera generación de móviles, denominada 3G, evoluciona para integrar todos los servicios ofrecidos por las distintas tecnologías y redes actuales, como GSM, TACS, DECT, RDSI e Internet, utilizando cualquier tipo de terminal, sea un teléfono fijo, inalámbrico o móvil, tanto en un ámbito profesional como doméstico, ofreciendo una mayor calidad de los servicios y soportando la personalización por el usuario y los servicios multimedia móviles en tiempo real. La velocidad de transferencia de datos que la UIT requiere en su solución IMT-2000 va desde los 144 kbit/s en vehículos a gran velocidad hasta los 2 Mbit/s en terminales en interiores de edificios (cifra al menos 60 veces superior a la que se tenía hasta hace poco utilizando un módem vocal y la RTC), pasando por los 384 kbit/s para usuarios móviles en el extrarradio, o vehículos a baja velocidad. En la siguiente figura se ilustra una de las aspiraciones de los servicios 3G: el Roaming Internacional.

Figura 2. ***Roaming internacional entre redes***

Los servicios 3G combinan el acceso móvil de alta velocidad con los servicios basados en el protocolo IP. Pero esto no sólo conlleva una conexión rápida con la World Wide Web, sino que implica además nuevas formas de comunicarse, de acceder a la información, de hacer negocios, de aprender y de disfrutar del tiempo libre, dejando a un lado las conexiones lentas, los grandes terminales y los puntos de acceso fijos. Con la 3G se pueden realizar múltiples conexiones simultáneamente desde un mismo terminal móvil. Así, por ejemplo, un usuario podría conectarse a una base de datos remota para obtener información sin necesidad de interrumpir una sesión de videoconferencia.

Para que los usuarios utilicen los servicios 3G hará falta nuevos teléfonos y otros dispositivos capaces de proporcionar los servicios que se deseen, desde los de telefonía móvil hasta los de multimedia (voz, datos y vídeo). Además, en las redes móviles es necesario introducir nuevos sistemas de transmisión por radio, cambiar parte de las plataformas de conmutación y de transmisión, e incorporar los nodos de servicio que hagan posibles las prestaciones 3G.

Si el paso de los sistemas de primera generación (analógicos) a los de segunda consistió, básicamente, en el cambio de la tecnología, el paso de la segunda a la tercera generación significará además el **cambio de modelo de negocio**. En este nuevo modelo las aplicaciones personalizadas/localizadas y multimedia serán las protagonistas, tanto para los ciudadanos residenciales como para los empresariales, viendo los operadores como se incrementa el tráfico en sus redes y se crean nuevas oportunidades de negocio.

Existen, además, razones evidentes que explican la necesidad de introducir la 3G. Por una parte se encuentra la capacidad de las redes móviles actuales, que sólo permiten albergar un número determinado y limitado de usuarios, con un patrón de consumo similar al actual, y que en cuanto se sobrepasa la congestión de la red se manifiesta de manera insoportable para los usuarios. Por otra parte, se encuentra el incremento de tráfico motivado por la sustitución del tráfico fijo por el móvil, en cuanto el coste de las llamadas se reduzca y los hábitos de los usuarios se modifiquen, necesitándose entonces más espectro. Y, por último, la aparición de nuevos servicios, muchos de ellos personalizados, donde la convergencia con Internet y el aumento de aplicaciones multimedia supondrán un aumento significativo de tráfico; tanto es así que los analistas estiman que supondrá en torno a un 30% de tráfico adicional en tan sólo dos o tres años.

Los fabricantes de infraestructura y terminales móviles están haciendo un gran esfuerzo para disponer de los equipos de 3G, ya que ven en ella una gran oportunidad para el desarrollo de su negocio y supervivencia a largo plazo. Así, todos los grandes, como Alcatel, Ericsson, Lucent, Motorola, Nokia, Nortel o Siemens, enfocan su estrategia en esta línea y se esfuerzan por desarrollar los estándares y fabricar los equipos para que estén a tiempo, además de impulsar el desarrollo de aplicaciones mediante la alianza con terceros.

1.2 IMT (International Mobil Telecomunications)

IMT-2000 es una norma de la ITU para los sistemas de la 3G que proporcionará acceso inalámbrico a la infraestructura de telecomunicaciones global por medio de sistemas satelitales y terrestres, para dar servicio a usuarios fijos y móviles en redes públicas y privadas en siglo XXI.

1.2.1 Objetivos de IMT-2000

Los objetivos primarios de ITU para IMT-2000 son:

- La eficacia operacional, particularmente para los datos y servicios multimedia.
- Flexibilidad y transparencia en la provisión de servicio global.
- La tecnología adecuada para aumentar la penetración de las telecomunicaciones, es decir, ofrecer un coste accesible para millones de personas en el mundo que todavía no tienen teléfono.
- La incorporación de toda una variedad de sistemas.
- Alto grado de uniformidad de diseño a escala mundial.
- Alto nivel de calidad, comparable con la de una red fija.
- Utilización de un terminal de bolsillo a escala mundial.
- La conexión móvil-móvil y móvil-fijo.
- La prestación de servicios por más de una red en cualquier zona de cobertura.

1.2.2 Requisitos de un sistema de tercera generación

- Alta velocidad en transmisión de datos: hasta 144 Kbit/s sobre vehículos; hasta 384 Kbit/s en peatones y hasta 2 Mbit/s en terminales estáticos.
- Transmisión de datos simétrica y asimétrica.
- Servicios de conmutación de paquetes y en modo circuito, tales como tráfico Internet (IP) y video en tiempo real.
- Calidad de voz comparable con la calidad ofrecida por sistemas cableados.
- Mayor capacidad y mejor eficiencia del espectro con respecto a los sistemas actuales.
- Capacidad de proveer servicios simultáneos a usuarios finales y terminales.
- Incorporación de sistemas de segunda generación y posibilidad de coexistencia e interconexión con servicios móviles por satélite.
- Itinerancia internacional entre diferentes operadores (Roaming Internacional).

Los sistemas de tercera generación deberán proveer soporte, entre otras, a aplicaciones como:

- Servicios unificados de mensajes como correo electrónico multimedia.

- Aplicaciones de comercio electrónico móvil, que incluye operaciones bancarias y compras móviles.
- Aplicaciones audio/video en tiempo real como videoteléfono, videoconferencia interactiva, audio y música, aplicaciones multimedia especializadas como telemedicina y supervisión remota de seguridad.

1.3 Los factores de éxito de 3G

El lanzamiento de un nuevo producto, servicio o tecnología se enfrenta siempre a enormes incertidumbres. Son muchos los casos de tecnologías muy prometedoras que, tras impulsar grandes inversiones de capital y talento, no fueron adoptadas por los mercados en la escala que inicialmente se había previsto. Los servicios 3G han despertado grandes expectativas y muchas compañías han apostado parte de su futuro en el despliegue y lanzamiento de los servicios de tercera generación.

La existencia de países como Japón o Corea, u otros en entornos más próximos donde los servicios 3G llevan ya años desplegados, permiten contar con excelentes bancos de pruebas que permiten reducir los riesgos, desechando los modelos menos exitosos y aprender de las estrategias que alrededor de 3G han gozado de mejor acogida por parte de los usuarios.

En los distintos mercados analizados parece que hay un patrón común a las empresas que han tenido éxito en el lanzamiento del servicio de banda ancha en movilidad. En sus estrategias han primado: la cobertura, la disponibilidad de terminales y el lanzamiento de nuevos servicios capaces de aprovechar al máximo las nuevas oportunidades de 3G.

- **La cobertura**

  El despliegue de las redes tiene mucho que ver con el esfuerzo inversor. Por ejemplo, Vodafone España fijó su plan de inversión en 2.880 millones de euros en España desde abril de 2004 a marzo de 2009 (inversión que incluye tanto la ampliación de capacidad de los servicios 2G como el despliegue de 3G). Este

esfuerzo inversor se puede, sin embargo, ver lastrado por las dificultades normativas que imposibilitan la creación de la infraestructura.

- **La disponibilidad de terminales**

Pocos sectores de actividad ofrecen un grado de competencia similar, que se manifiesta en un gran esfuerzo por parte de las compañías por entender qué combinación de elementos es la que más valora un cliente a la hora de optar por una de las operadoras que actualmente compiten en España. En un reciente estudio, se pone de manifiesto que las principales razones para cambiar de operador son, por este orden: el cambio de terminal, la complejidad de las tarifas y la calidad del servicio. La enorme relevancia que el usuario español da al cambio del teléfono móvil lleva al conjunto de las operadoras a invertir anualmente en el subsidio de los terminales entre 850 y 1.000 millones de euros. La disponibilidad de terminales con las prestaciones adecuadas para sacar partido de los nuevos servicios es uno de los grandes retos del sector a corto y medio plazo.

- **Los nuevos servicios**

El lanzamiento de los primeros servicios 2,5G (GPRS y EDGE) para particulares y empresas ofreció a los clientes la posibilidad de familiarizarse con los servicios de datos en movilidad. La introducción de la banda ancha en el móvil permite ofrecer servicios multimedia más avanzados -especialmente de audio y video-, abrir nuevas formas de comunicación -como la videoconferencia- o agilizar el acceso a la información empresarial - tarjeta de datos 3G-; pero antes del lanzamiento, cientos de miles de usuarios estaban ya familiarizados con nuevas formas de utilizar la telefonía móvil. Esta base de usuarios es uno de los principales activos para el lanzamiento de la tercera generación.

La oportunidad que tenemos por delante es formidable, la nueva comunicación móvil se puede afianzar como una de las principales plataformas para el desarrollo de la Sociedad de la Información en España. El éxito en el

despliegue del servicio depende, naturalmente y en primer lugar, de la capacidad de las operadoras para saber convertir la nueva tecnología en productos y soluciones atractivos para los usuarios; este esfuerzo demanda un enorme esfuerzo inversor. Pero, no menos importante, es la necesidad de que exista un entorno regulador favorable, poco intervencionista y estable que estimule la inversión. En el caso español es, además, especialmente necesario que la mesa de concertación, impulsada desde la Administración Central, tenga éxito y sirva para eliminar los obstáculos que impiden la instalación de nuevas estaciones base que han de garantizar la correcta prestación del servicio. La telefonía móvil es un activo del conjunto de la sociedad y puede ser uno de los grandes dinamizadores de la Sociedad de la Información, de la capacidad de las operadoras y de las Administraciones Públicas para armonizar sus intereses y aunar esfuerzos siempre que, como país, estemos a la altura del reto planteado.

En realidad lo más sobresaliente de la telefonía móvil debería ser su aportación a la Sociedad de la Información , extendiendo la información y el ocio al ciudadano, acercando los sistemas de información a empresas, aumentando así su productividad y lo que desde un punto de vista social resultaría más importante, integrar a todas y cada una de las comunidades existentes en la sociedad en un espacio, donde no sólo se garantice la información a las personas y comunicación entre ellas, se fomente la productividad entre las empresas y la relación de éstas con las Administraciones, si no que, y esto es mucho más ambicioso, se desarrolle la convivencia y la tolerancia entre las personas y las ideas, en un mundo que no puede permitirse la exclusión por tanto de valor o valores no productivos en sectores marginados o en desigualdad.

1.4 El Foro UMTS

En Europa, el Instituto Europeo de Telecomunicaciones (ETSI) propuso la norma paneuropea de tercera generación UMTS (Universal Mobile Telecommunications System). UMTS es miembro de la familia global IMT-2000 del sistema de comunicaciones móviles de tercera generación de UIT. El estándar UMTS (Universal

Mobile Telephone System) está basado en la tecnología W-CDMA. UMTS está gestionado por la organización 3GPP, también responsable de GSM, GPRS y EDGE.

En la implantación de los sistemas 3G juega un papel importantísimo el Foro UMTS (www.umts-forum.org), un organismo independiente creado en diciembre de 1996 en el que participan casi 170 compañías de 30 países pertenecientes a las industrias suministradoras de equipos, operadores de telecomunicaciones y organismos de regulación. El Foro está comprometido en la formación del consenso necesario para introducir y desarrollar con éxito el estándar UMTS y así poder satisfacer la demanda del mercado de unas comunicaciones móviles personales de bajo coste y alta calidad. En la siguiente figura se puede observar los pasos que se han seguido y se siguen en la implantación de UMTS.

Figura 3. ***Evolución de GSM hacia UMTS***

El Foro UMTS también actúa como catalizador con las organizaciones especializadas que tratan sobre estandarización y espectro, entre otros aspectos, y mantiene relaciones con organizaciones de carácter regional y mundial, organismos de estandarización y otras comunidades reconocidas tanto de la industria como de operadores.

Una aportación, básica, pero esclarecedora e imprescindible del Foro es la propia definición de UMTS, dejando aparte sus aspectos tecnológicos, como "un sistema de comunicaciones móviles que ofrece significativos beneficios a los usuarios, incluyendo una alta calidad y servicios inalámbricos multimedia sobre una red

convergente con componentes fijos, celulares y por satélite. Suministrará información directamente a los usuarios y les proporcionará acceso a nuevos y novedosos servicios y aplicaciones. Ofrecerá comunicaciones personales multimedia al mercado de masas, con independencia de la localización geográfica y del terminal empleado (movilidad del terminal, personal y de servicios)".

Actualmente, del estándar 3G basado en la evolución de las redes GSM (G-UMTS) se ocupa 3GPP (Third Generation Partnership Project), /www.3gpp.org/, creado en diciembre de 1998 y formado por grupos de estandarización de todo el mundo. En concreto, participan ETSI (Instituto Europeo de Estándares de Telecomunicación) en Europa, TTC y ARIB (Asociación de las Empresas del Sector de Radio) en Japón, TTA (Asociación de Tecnologías de Telecomunicaciones) en Corea, T1 en Estados Unidos y CWTS en China, junto a diversos fabricantes y operadores. La finalidad del 3GPP consiste en establecer especificaciones abiertas aceptadas en todo el mundo para garantizar, entre otras cosas, la itinerancia mundial, mediante la cooperación entre distintos organismos de normalización nacionales y regionales.

En Estados Unidos, de manera similar, la TIA puso en marcha 3GPP2, con el objetivo de estandarizar la tecnología de acceso radio cdma2000, así como las interfaces hacia las redes centrales ANSI-41.

2. OSA/Parlay

2.1 Introducción

Esta sección ofrece una breve explicación sobre cómo se originó y desarrolló la Arquitectura de Servicios Abiertos (OSA). Al mismo tiempo introduce conceptos para ayudar a entender dicha arquitectura de servicios, como son el de Red Inteligente y CAMEL.

El concepto de arquitectura de Red Inteligente (IN) nació al final de los 80's como una extensión del Sistema de Señalización 7 (SS7). Esta arquitectura básicamente fue desarrollada para la red telefónica conmutada fija. La principal idea de la red inteligente es la de separar los servicios de control y las funciones de conmutación, con el objetivo de crear una arquitectura de servicios fácil de gestionar y mas eficiente en el momento de crear servicios para los usuarios.

La arquitectura de Red Inteligente es actualmente un estándar de la Unión de Telecomunicaciones Internacional - Sección Telecomunicaciones (ITU-T), el cual define elementos y relaciones que ofrecen como resultado servicios de valor añadido (AVS) en beneficio de los usuarios.

Por otro lado, la estandarización del Sistema Global de Comunicaciones Móviles, por sus siglas en ingles GSM (Global System for Mobile Communications), tuvo su inicio tiempo antes que la estandarización del modelo de Red Inteligente. Sin embargo, GSM fue desarrollado basándose en la red de digital de servicios integrados, por sus siglas en ingles "ISDN - Integrated Services Digital Network", pero tan pronto fue generado el estándar de Red Inteligente, se buscó relacionarlo con las redes móviles.

Para solucionar este problema, el Instituto Europeo de Estándares de Telecomunicación (ETSI European Telecommunications Standards Institute) definió Aplicaciones Adaptables para el enriquecimiento Lógico de Redes Móviles (CAMEL - Customized Applications for Mobile Network Enhanced Logic), las cuales describen una arquitectura similar a la de Red Inteligente pero aplicable a redes móviles.

CAMEL fue el primer intento para adaptar el concepto de Red Inteligente a Redes Móviles. Tiene la capacidad de proveer servicios de conmutación de voz debido a que está basado en la arquitectura de la Red Inteligente. Sin embargo, la aparición del Sistema Universal de Telecomunicaciones Móviles (UMTS – Universal Mobile Telecommunication System) ha hecho que CAMEL no sea óptimo para ser implementado. Por otro lado GSM es una red conmutada de voz genérica, entonces conceptualicemos a UMTS como una red total de integración de voz y datos, basada completamente en transporte y encaminamiento de datos. UMTS es la unión entre Internet, Redes Telefónicas, Redes Móviles y la Red de Transmisión de TV y radio. El modelo de servicio que ofrece CAMEL es muy restrictivo para este ambiente de servicios integrados.

Siguiendo estos principios, las líneas de trabajo se centraron en la integración de los conceptos de Red Inteligente y CAMEL, surgiendo así la Arquitectura de Servicios Abiertos (o también llamado Acceso a Sistemas Abiertos – *Open Service Access)*, la cual permite el aprovisionamiento de servicios a través de tecnologías y sistemas de gestión diferentes. Esta herramienta abre la posibilidad de establecer modelos independientes de negocios por los gestores de las redes de telecomunicaciones de tercera generación. OSA es la arquitectura para servicios móviles desarrollada por el 3GPP (*3rd Generation Partnership Program*) que también fue adoptada por el 3GPP2.

Actualmente Internet ha creado oportunidades para implementar nuevas comunicaciones y servicios, pero el modelo de Internet es totalmente opuesto a CAMEL y al concepto de Red Inteligente. Mientras que en la red inteligente los servicios son ofrecidos y gestionados por un operador de red, Internet abre la posibilidad para que cualquier entidad con infraestructura básica pueda ofrecer sus servicios. Parlay nace como la solución al problema de coexistencia entre estos modelos.

Parlay es la definición de una interfaz estándar y abierta para permitir el acceso a los recursos de la red a operadores y aplicaciones que no tengan que estar en el dominio del operador; es la especificación de la API de OSA.

En la actualidad, se utilizan los términos OSA, Parlay y OSA/Parlay indistintamente. OSA/Parlay es una API (*Application Programming Interface*) que permite la creación de servicios de telecomunicaciones de manera rápida. Está definida por el grupo Parlay, que es un consorcio de empresas del mundo de las telecomunicaciones y la informática con fines no lucrativos. Actualmente está compuesto por más de 65 compañías (Alcatel, HP, IBM, Lucent, Siemens, Sun, British Telecom, NTT, etc.) El grupo Parlay fue fundado en 1998 y desde entonces se han creado cuatro versiones de las especificaciones Parlay. La mayoría del trabajo técnico se realiza en grupos de trabajo que incluyen miembros de Parlay, 3GPP y ETSI.

La API de OSA/Parlay es independiente de la tecnología. Está diseñada para ser usada en redes móviles, fijas y redes de nueva generación basadas en protocolo IP. También es independiente del lenguaje de programación utilizado (Java, C, C++, etc.).

OSA/Parlay está basado en estándares abiertos tales como CORBA, IDL, Java, UML y Servicios Web (SOAP, XML y WSDL).

El objetivo del grupo Parlay cuando se fundó fue potenciar la convergencia de los mundos de las comunicaciones y la informática. Una manera de lograr esto es usar la tecnología que se utiliza para la creación de aplicaciones informáticas para desarrollar servicios de comunicaciones.

En los siguientes apartados se verá cuales son los beneficios de OSA, dónde encaja en la red, se describirá qué funcionalidades ofrece, los mecanismos principales y se hará un recorrido por la especificación.

2.2 Beneficios de OSA/Parlay

Los principales beneficios que se consiguen utilizando OSA/Parlay son los siguientes:

- Rapidez en la creación de servicios.
- Independencia de la red.
- Independencia del fabricante.
- Existencia de un gran número de desarrolladores de aplicaciones.
- Vendedores de aplicaciones y servicios basados en OSA/Parlay independientes.

2.2.1 Rapidez en la creación de servicios

OSA/Parlay está basada en las mismas soluciones que los desarrolladores de software comercial usan cuando construyen aplicaciones, al contrario que los entornos de desarrollo propietarios para la creación de servicios que se estaban usando. Una ventaja de esto es que es muy fácil desarrollar aplicaciones OSA/Parlay usando herramientas de desarrollo normales para aplicaciones Java tales como el J-Builder de Borland o el Websphere Studio de IBM, incluso con entornos gratuitos. Muchos vendedores como Ericsson y Lucent Technologies también proporcionan herramientas de simulación, pudiendo simular las aplicaciones desde ordenadores personales.

Ericsson, IBM y Telenor han realizado experimentos que demuestran que usando OSA/Parlay se reducen significativamente los tiempos de desarrollo de servicios de comunicación.

2.2.2 Independencia de la red

La API de OSA/Parlay fue diseñada, en la medida de lo posible, para que fuera independiente de la red que hubiera debajo. Una aplicación móvil no necesita saber qué

tecnología se está usando para implementar los servicios de localización. El desarrollador de la aplicación tan solo necesita saber las coordenadas de la localización y la precisión. Esto permite a una aplicación funcionar igual de bien en diferentes redes, y la misma aplicación se puede instalar incluso en redes fijas. De nuevo, la aplicación no se preocupa de cómo se determina la información de localización, sólo de los datos de localización. Las mismas consideraciones se pueden aplicar a otras áreas funcionales, tales como establecer una llamada, cargar al usuario una cierta cantidad por el uso del servicio, etc.

2.2.3 Independencia del fabricante

Una consecuencia de la independencia de la tecnología (y la red) de OSA/Parlay es que también son independientes del fabricante. Los proveedores de servicios se benefician al tener un conjunto único de interfaces que son soportadas por muchos fabricantes de plataformas. Esto añade aún más flexibilidad en el desarrollo.

2.2.4 Existencia de un gran número de desarrolladores de aplicaciones

El uso de tecnología estándar software significa que los desarrolladores de software con experiencia en C++ y CORBA o Java pueden desarrollar aplicaciones usando OSA/Parlay fácilmente. Se estima que hay entorno a 1,5 millones de desarrolladores de C y Java en el mundo en contraste con los desarrolladores de servicios en RI que necesitan un entrenamiento muy especializado y no hay más que unos pocos miles en el mundo.

2.2.5 Vendedores de aplicaciones y servicios basados en OSA/Parlay independientes

Un resultado del uso de tecnología de desarrollo de software estándar es que se está creando una comunidad creciente de vendedores de software independientes que están desarrollando y vendiendo aplicaciones y servicios basados en OSA/Parlay. El grupo Parlay realiza cada cierto tiempo demostraciones donde los vendedores pueden demostrar las nuevas aplicaciones y servicios basados en OSA/Parlay.

2.3 Arquitectura de OSA

Los servicios de 3G no se basan en servicios tradicionales detallados como los servicios suplementarios de los sistemas 2G, sino que proporcionan servicios usando herramientas de desarrollo genéricas, como OSA.

En OSA los servicios se implementan con aplicaciones que usan los recursos de la red accesibles a través de las interfaces de OSA.

OSA puede dividirse en tres partes:

- **Aplicaciones**: implementadas en servidores de aplicación. Ejemplos de aplicaciones son tarificación por contenidos, conferencia, aplicaciones basadas en localización, etc.
- **Armazón (*Framework*)**: proporciona los mecanismos básicos para que las aplicaciones puedan usar los recursos de la red. Ejemplos de estos mecanismos son la autentificación y el descubrimiento de funcionalidades.
- **Servidores de Capacidades de Servicio (SCS)**: contienen los conjuntos de capacidades de servicio denominados Funcionalidades de Capacidades de Servicio (SCF – *Service Capability Features*), que son abstracciones de las funcionalidades de la red real.

Los SCF's de OSA se definen a partir de sus interfaces y métodos. Las interfaces se dividen en dos grupos:

- Interfaces de Armazón.
- Interfaces de Red.

2.4 Mecanismos básicos de OSA

Se pueden clasificar según las entidades involucradas:

- **Entre Aplicación y Armazón**
- **Entre Armazón y SCS**
- **Entre Servidor de aplicaciones y SCS**

Algunos sólo se realizan una vez y otros cada vez que un usuario se subscribe a la aplicación.

2.4.1 Mecanismos Básicos entre Aplicaciones y Armazón

- **Autentificación**

Una aplicación puede acceder a la función de autentificación si previamente existe un acuerdo de servicio. La aplicación debe autentificar al Armazón y viceversa. Esto debe realizarse siempre antes de poder usar otras funciones de OSA.

- **Autorización**

Consiste en determinar qué acciones puede realizar una aplicación previamente autentificada. Una aplicación podrá acceder a un determinado número de SCF's.

- **Descubrimiento de las funciones del Armazón y de los SCF's disponibles**

Después de realizar la autentificación correctamente, las aplicaciones pueden obtener qué funciones están disponibles en el Armazón y usar la función de Descubrimiento para obtener información de para qué SCF's tiene autorización. La función de Descubrimiento puede ser utilizada en cualquier momento después de la autentificación.

- **Establecimiento de acuerdos de servicio**

Antes de que una aplicación pueda interactuar con un SCF de red, hay que establecer un acuerdo de servicio. Éste constará de una parte "*offline*" (intercambio de documentos físicos por ejemplo) y una parte "*online*" (documentos electrónicos). La aplicación debe firmar los documentos electrónicos del acuerdo de servicio para poder acceder a los SCF's.

- **Acceso a los SCF's de red**

El Armazón proporciona funciones para controlar el acceso a SCF's o datos de servicio para cualquier método de la API desde cualquier aplicación, con niveles de seguridad adecuados.

2.4.2 Mecanismos Básicos entre Armazón y SCS

- **Registro de SCF's de Red**

Los SCF's que ofrece un SCS pueden ser registrados en el Armazón. Así, éste podrá informar a las aplicaciones cuando soliciten un listado de los SCF's disponibles. Este mecanismo se usa cuando se instala o actualiza un SCS.

2.4.3 Mecanismos Básicos entre Servidor de Aplicaciones y SCS

- **Petición de notificación de eventos**

Los SCF's que ofrece un SCS pueden ser registrados en el Armazón. Así, éste podrá informar a las aplicaciones cuando soliciten un listado de los SCF's disponibles. Este mecanismo se usa cuando se instala o actualiza un SCS.

2.5 Seguridad relacionada con el usuario final

Después de que una aplicación haya sido autentificada y autorizada para usar SCF's, los aspectos de seguridad relacionados con el usuario final suponen un problema importante. Existen tres aspectos a tener en cuenta que se enumeran a continuación.

2.5.1 Autorización a los usuarios finales para usar aplicaciones

Un usuario final estará autorizado a usar una aplicación sólo cuando se suscriba a ella. La suscripción es parte del Acuerdo de Nivel de Servicio (*SLA – Service Level Agreement*) que se firma entre el HE (*Home Environment*) y el HE-VASP (*Home Environment – Value Add Service Provider*) si el usuario final se ha suscrito a la aplicación antes de que la aplicación acceda a las SCF's.

El HE controla las suscripciones y puede usar políticas para definir y restringir la lista de servicios ofrecidos al usuario final. El HE tiene que compartir la información

de suscripción y activación de servicios con el HE-VASP para que éste conozca qué usuarios pueden usar sus servicios.

2.5.2 Autorización a las aplicaciones de los usuarios finales

El HE podrá proporcionar capacidades de servicio a una aplicación si las siguientes condiciones se cumplen:

- El usuario final se ha suscrito a la aplicación.
- El usuario final ha activado la aplicación.
- El uso de esa capacidad de servicio de red no viola la privacidad del usuario final.

Cada vez que una aplicación intenta usar un SCF para un determinado usuario final, el SCS se asegura de que las anteriores condiciones se cumplen.

2.5.3 Opciones de privacidad del usuario final

El HE puede ofrecer al usuario final un conjunto de opciones de privacidad. Por ejemplo, puede permitir decidir si la posición del usuario puede darse a terceras partes, o si el usuario acepta recibir información no solicitada expresamente a su terminal. El SCS asegura que la aplicación no viola estas preferencias.

2.6 OSA/Parlay en la Red de Telecomunicaciones

El modelo de OSA/Parlay añade un nuevo elemento en la red, la pasarela o plataforma OSA/Parlay, usada para enlazar las aplicaciones que usan la API de OSA/Parlay con los elementos de red existente. Estará bajo el control del operador de red o del proveedor de servicio y es un punto por el que pasarán todas las interacciones OSA/Parlay. Es aquí donde residirá el Armazón. Gracias a esto las aplicaciones son independientes de los protocolos específicos usados en la red, permitiendo que las redes evolucionen y cambien sin afectar a los servicios y aplicaciones. En la figura siguiente se puede observar la arquitectura típica de OSA/Parlay.

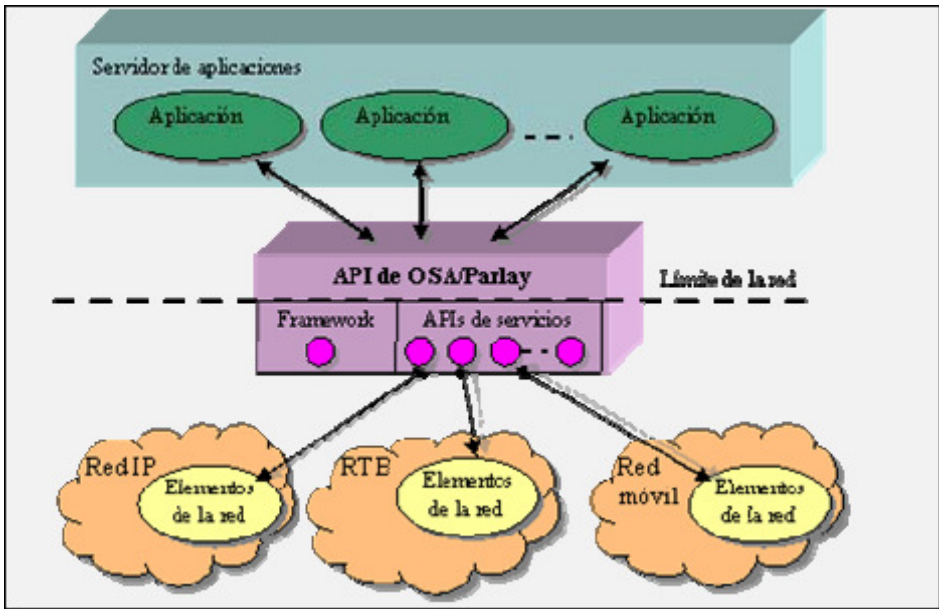

Figura 4. ***Arquitectura típica de OSA/Parlay***

2.7 Especificación de OSA/Parlay

Se lista a continuación el contenido de la especificación de OSA/Parlay (versión 4.1), que contiene la definición de todas las SCF's disponibles en el SCS. Las especificaciones han sido definidas conjuntamente entre ETSI, el grupo Parlay y 3GPP en colaboración con un conjunto de compañías miembros de la comunidad JAIN.

El documento completo denominado "*Open Service Access (OSA); Application Programming Interface (API)"* consta de las siguientes partes:

- Parte 1: Introducción (*Overview*)
- Parte 2: Definiciones de Datos Comunes (*Common Data Definitions*)
- Parte 3: **Armazón** (*Framework*)
- Parte 4: **SCF Control de Llamada** (*Call Control*)
    - Subparte 1: Definiciones Comunes del Control de Llamada (*Call Control Common Definitions*)
    - Subparte 2: SCF Control de Llamada Genérico (*Generic Call Control SCF*)

  - Subparte 3: SCF Control de Llamada Multi-Partícipe (*Multi-Party Call Control SCF*)
  - Subparte 4: SCF Control de Llamada Multimedia (*Multi-Media Call Control SCF*)
  - Subparte 5: SCF Control de Llamada de Conferencia (*Conference Call Control SCF*)
- Parte 5: **SCF Interacción de Usuario** (*User Interaction SCF*)
- Parte 6: **SCF Movilidad** (*Mobility SCF*)
- Parte 7: **SCF Capacidades de Terminal** (*Terminal Capabilities SCF*)
- Parte 8: **SCF Control de Sesión de Datos** (*Data Session Control SCF*)
- Parte 9: **SCF Mensajería Genérica** (*Generic Messaging SCF*)
- Parte 10: **SCF Gestor de Conectividad** (*Connectivity Manager SCF*)
- Parte 11: **SCF Gestión de Cuentas** (*Account Management SCF*)
- Parte 12: **SCF Tarificación** (*Charging SCF*)
- Parte 13: **SCF Administración de Políticas** (*Policy Management SCF*)
- Parte 14: **SCF Gestión de Presencia y Disponibilidad** (*Presence and Availability Management SCF*)

Las especificaciones de OSA definen una arquitectura que permite a los desarrolladores de servicios hacer uso de las funcionalidades de la red a través de una interfaz estándar abierta. Las funcionalidades de red se describen como Funcionalidades de Capacidades de servicio (SCF – *Service Capability Features*) o simplemente Servicios (ambos términos se utilizan indistintamente en la especificación). El Armazón de OSA es el componente que conecta los Servicios con las Aplicaciones.

El propósito de la API de OSA es encapsular la complejidad de la red, sus protocolos y su implementación específica de las aplicaciones. Las aplicaciones no tienen que conocer qué nodos de la red interactúan con un SCS para proporcionar una SCF a la aplicación. La red específica subyacente y sus protocolos son transparentes a la aplicación.

La primera parte de la especificación contiene una introducción, la segunda parte contiene las definiciones de los tipos de datos comunes y la tercera parte las interfaces del Armazón. Las siguientes partes contienen las descripciones de los SCF's.

Cada una de las partes está estructurada de la siguiente manera:

- Diagramas de secuencia
- Relaciones de clases
- Especificación de la interfaz
- Diagramas de transición de estados (STD – *State Transition Diagrams*)
- Definición de tipos de datos

La API de OSA está definida usando UML que es independiente de la tecnología. OSA se puede realizar de múltiples maneras y además de la definición en UML, la especificación incluye:

- Un anexo normativo con la API de OSA en IDL que especifica la realización en CORBA.
- Un anexo informativo con la API de OSA en WSDL que especifica la realización en SOAP/HTTP.
- Un anexo informativo con la API de OSA en Java (conocida como "JAIN *Service Provider* API).

A continuación se van a detallar a modo de ejemplo las partes 4, 5 y 6 de la especificación, es decir, la de control de Llamada (*Call Control SCF*), Interacción de Usuario (*User Interaction SCF*) y Movilidad (*Mobility SCF*), por ser tres de las SCF's más usadas en la aplicación desarrollada en este proyecto.

2.7.1 SCF Control de Llamada

En la norma ES 202 915, Parlay 4, se especifican 4 partes de la API Control de Llamada (CC – *Call Control*):

- SCF Control de Llamada Genérico (GCC - *Generic Call Control SCF*)
- SCF Control de Llamada Multi-Partícipe (MPCC - *Multi-Party Call Control SCF*)

- SCF Control de Llamada Multimedia (MMCC - *Multi-Media Call Control SCF*)
- SCF Control de Llamada de Conferencia (CCC - *Conference Call Control SCF*)

Tres de éstas se han incluido en la versión 5 del 3GPP: Control de Llamada Genérico, Control de Llamada Multi-Partícipe y Control de Llamada Multimedia. GCC es la misma API que estaba presente en la previa especificación para 3GPP Versión 99 (TS 129 198 V3). MPCC se introdujo en la versión 4 y MMCC en la versión 5. Las 4 estaban ya incluidas en ETSI 201 195, Parlay 3.

El trabajo conjunto entre los grupos 3GPP CN5, ETSI SPAN y el grupo de trabajo de control de llamada de Parlay con la colaboración de JAIN se han centrado en MPCC y MMCC. Se han mejorado mucho las funcionalidades y para ello ha sido necesario eliminar la herencia que había entre GCC y MPCC. Se ha dejado de trabajar en GCC y la futura base del control de llamada es ahora MPCC.

El modelo de llamada usado en esta SCF tiene los siguientes objetos:

- Un objeto **llamada**. Una llamada es la relación entre varios participantes. Está relacionado con la visión de la llamada como un todo que tiene la aplicación.
- Un objeto **extremo de llamada** (*call leg*). Representa la asociación lógica entre una llamada y una dirección. La relación incluye al menos la relación de señalización con el participante. La relación con la dirección se hace cuando el extremo es encaminado. Antes el objeto estará "inactivo" y no asociado aún con la dirección.
- Una **dirección**. Representa de forma lógica a un participante en la llamada.
- Un **terminal**. Es el punto final de la señalización y/o datos para un participante. Este objeto no se considera en la especificación.

El objeto llamada se usa para establecer una relación entre varios participantes creando un extremo de llamada para cada participante. Junto con la relación de señalización que representa un extremo de llamada puede haber una conexión portadora

(en redes tradicionales de sólo voz) o un determinado número (cero o más) de canales de medios (en redes multimedia).

Un extremo de llamada puede vincularse a una llamada o desvincularse. Cuando un extremo está vinculado, quiere decir que los canales de información (portadores o multimedia) del extremo están conectados a los canales de información de los otros extremos vinculados a la misma llamada. Normalmente existe un número limitado de extremos que pueden estar siendo encaminados o conectados a la llamada.

Algunas redes distinguen entre extremos de llamada pasivos y controladores. Por definición la llamada será liberada cuando el extremo controlador sea liberado. El resto de extremos se denominan pasivos. Actualmente, no existe manera de especificar que extremo es el controlador.

Existen dos formas en las que la aplicación obtiene el control de una llamada. La aplicación puede solicitar ser notificada de las llamadas que cumplen unos determinados criterios. Cuando una llamada que cumpla esos criterios se produzca, se informa a la aplicación que puede pasar a controlar la llamada. Otra forma es crear una nueva llamada desde la propia aplicación.

A continuación se describirán cada una de las subpartes de esta API.

2.7.1.1 SCF Control de Llamada Genérico

El Servicio de Control de Llamada Genérico (GCCS) proporciona el servicio de control de llamada básico a la API. Está basado en un modelo de tres participantes, que permite que las llamadas puedan ser creadas por la red y encaminadas a través de la red.

El GCCS soporta las suficientes funcionalidades para permitir el encaminamiento de llamadas y la gestión de llamadas para los servicios de Red Inteligente actuales en el caso de redes telefónicas conmutadas, o su equivalente en redes basadas en paquetes.

La intención del GCCS es estar preparada para especificaciones de control de llamada tales como las recomendaciones de la ITU-T H.323, Q.763 ISUP, Q.931 y Q.2931, la especificación UNI3.1 del ATM Forum y la recomendación del IETF RFC 3261 SIP (*Session Initiation Protocol*) o cualquier otra tecnología de control de llamada.

Aquí las llamadas están limitadas a llamadas de dos participantes. No se puede acceder explícitamente a los extremos de la llamada ni a los medios. De esto se encargan MPCC y MMCC.

El GCCS está representado por las clases "IpCallControlManager" e "IpCall". Algunos métodos son asíncronos y no bloquean mientras se realizan las tareas. Por tanto, el cliente puede manejar muchas más llamadas que si se usaran llamadas síncronas. Para manejar las respuestas y los informes, los desarrolladores deben implementar dos clases "IpAppCallControlManager" e "IpAppCall". En las siguientes figuras se muestra el diagrama de clases.

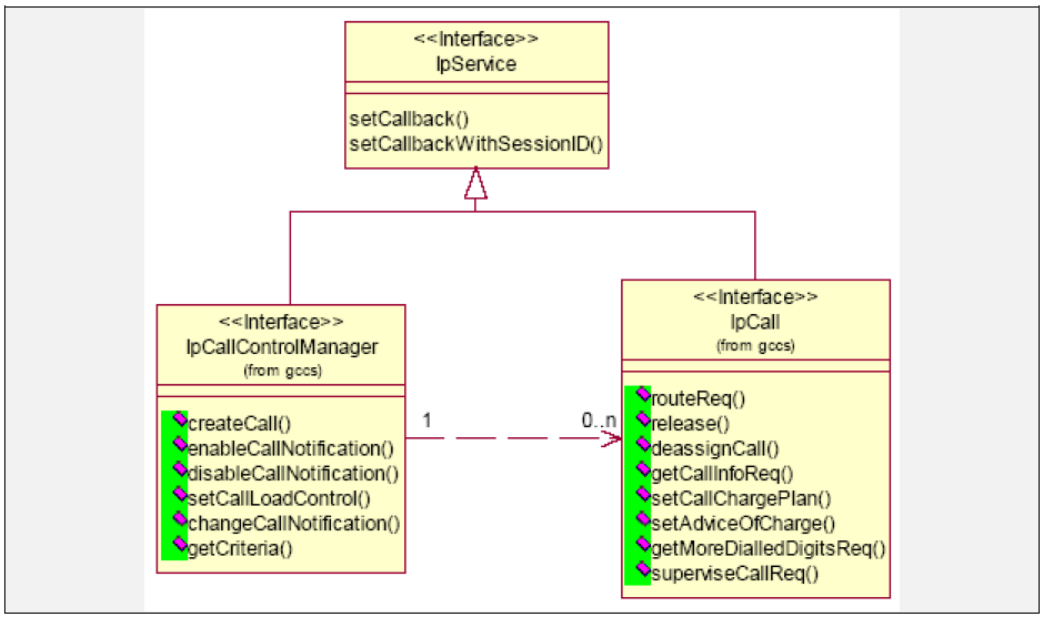

Figura 5. ***GCC. Interfaces del servicio***

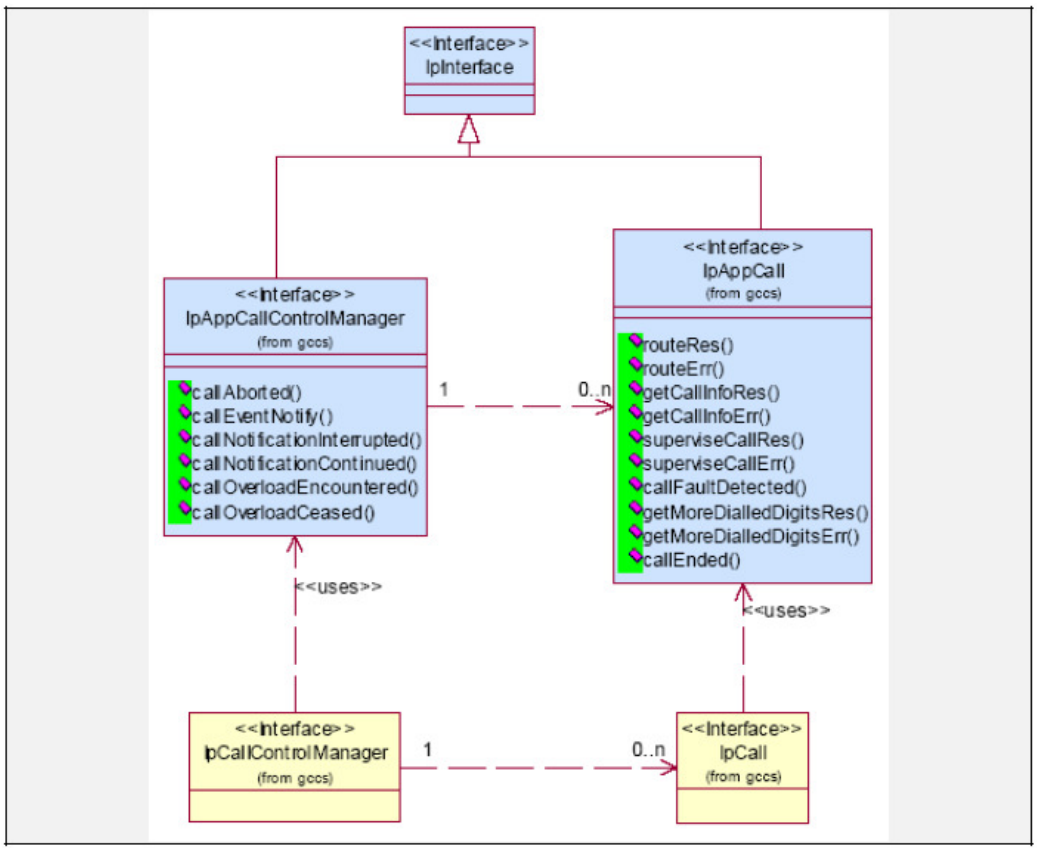

Figura 6. ***GCC. Interfaces de la aplicación***

2.7.1.2 SCF Control de Llamada Multi-Partícipe

Este servicio mejora la funcionalidad del GCCS con la gestión de extremos de llamada. Permite también el establecimiento de llamadas multi-partícipe, conectando muchos extremos simultáneamente a una misma llamada.

El servicio MPCC está formado por las clases "IpCallLeg", "IpMultiPartyCall" e "IpMultiPartyCallControlManager" que sirven de interfaz a los servicios de la red. Algunos métodos son asíncronos igual que ocurría en GCCS para permitir controlar más llamadas simultáneamente. El desarrollador tiene que implementar las clases de *callback* para manejar las respuestas e informes. En la siguiente figura se muestra un diagrama de clases.

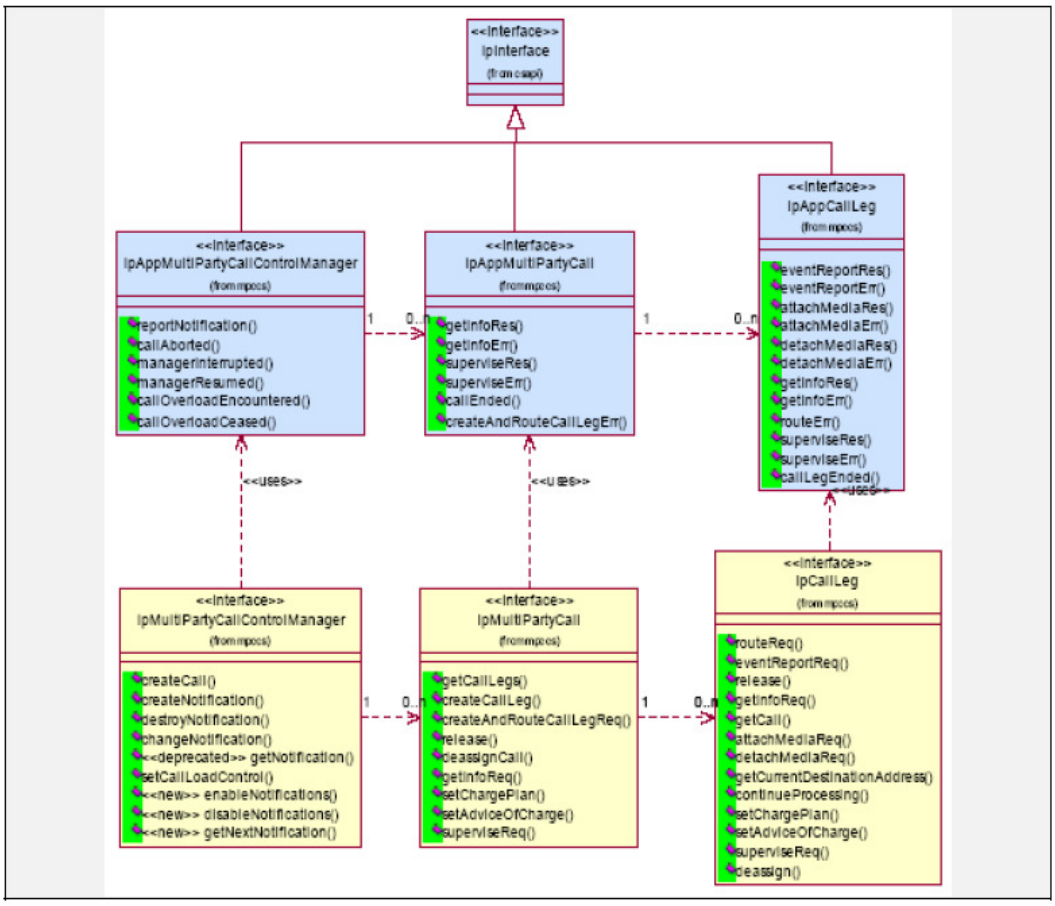

Figura 7. ***Paquete MPCC***

2.7.1.3 SCF Control de Llamada Multimedia

El servicio de Control de Llamada Multimedia amplía la funcionalidad del servicio MPCC con capacidades multimedia. Para manipular los aspectos multimedia de una llamada se introduce el concepto de **flujo de medios**. Un flujo de medios es un flujo bidireccional de medios asociado con un extremo de la llamada. Normalmente estos flujos de medios son negociados por los terminales de la llamada. La aplicación puede controlar estos flujos de las siguientes maneras:

- Activándose cuando se establece un flujo de medios que cumple unas características concretas definidas por la aplicación.
- Monitorizando el establecimiento (adicción) o liberación (substracción) de flujos de medios en una llamada saliente.
- Permitiendo o denegando el establecimiento de flujos de medios.

- Eliminando explícitamente flujos de medios ya establecidos.
- Solicitando el flujo de medios asociado a un determinado extremo de la llamada.

Al igual que en los dos últimos SCF's aquí existen operaciones asíncronas por el mismo motivo que antes. En las siguientes figuras se muestran las clases que componen este SCF.

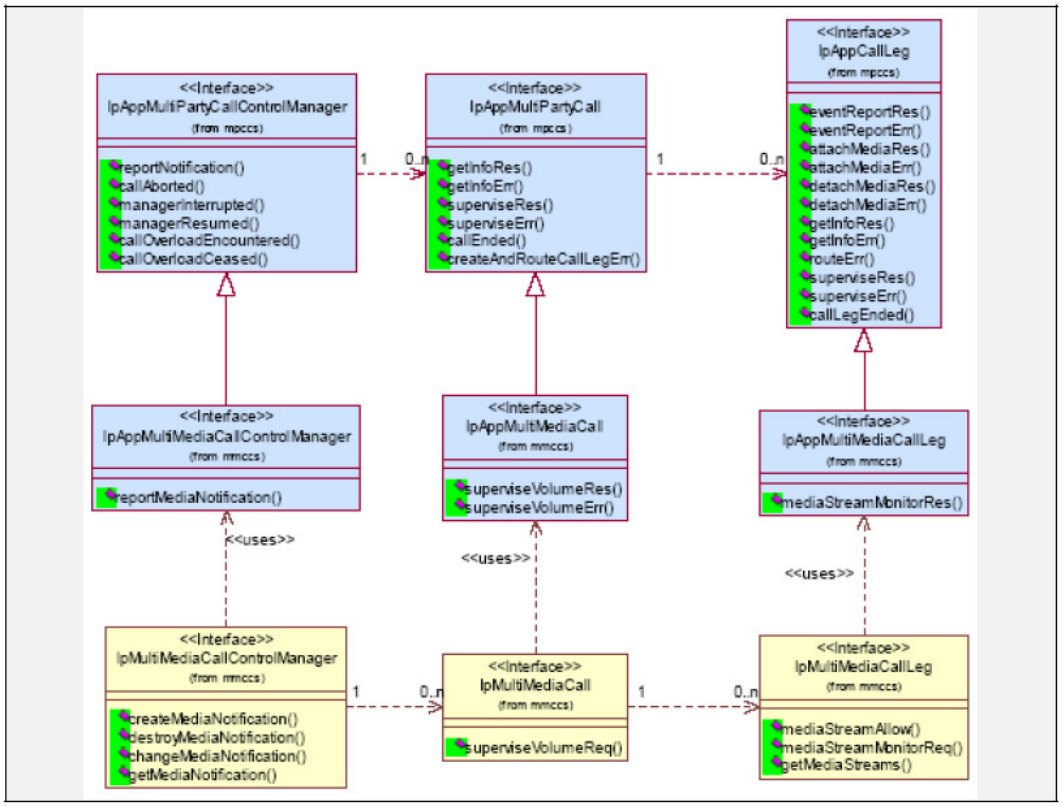

Figura 8. ***MMCC. Clases de la aplicación***

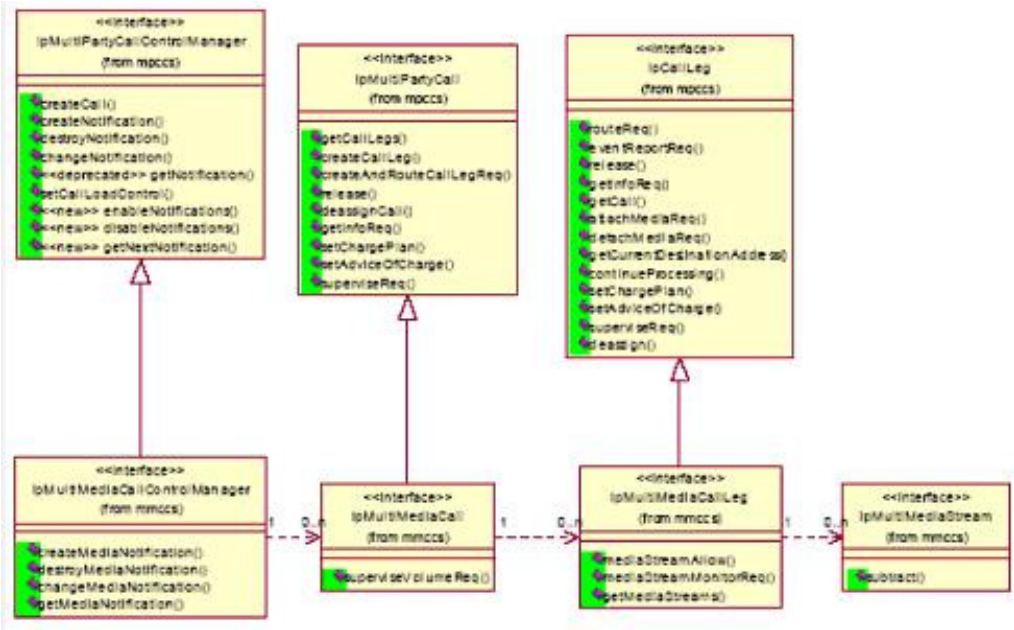

Figura 9. ***MMCC. Clases del servicio***

2.7.1.4 SCF Control de Llamada de Conferencia

El servicio Control de Llamada de Conferencia mejora el servicio MMCC. Da a la aplicación la posibilidad de manipular subconferencias dentro de una conferencia. Una subconferencia es un grupo de extremos de llamada dentro de una conferencia. Sólo los participantes de la misma subconferencia tienen una conexión de portadora (o una conexión de canales de medios) con los otros. Las aplicaciones pueden:

- Crear nuevas subconferencias dentro de una conferencia, bien como una subconferencia vacía o dividiendo una subconferencia existente en dos.
- Mover extremos de llamada entre subconferencias.
- Unir subconferencias en una.
- Obtener un listado de todas las subconferencias de la llamada.

La conferencia genérica también posibilita manipular detalles de una conferencia multimedia típica como:

- Interfuncionamiento con protocolos de conferencia (por ejemplo H.323).
- Manipulación de los medios en el MCU (difusión de video por ejemplo).

- Manejo de políticas de conferencia multimedia (cómo debe manipularse el video, cómo se controla la voz, etc.)

Pero además, este servicio añade el soporte para la reserva de recursos necesarios para realizar la conferencia. La aplicación puede:

- Reservar recursos durante un periodo de tiempo predefinido.
- Liberar recursos reservados.
- Buscar la disponibilidad de recursos de conferencia basándose en unos determinados criterios.

Existen dos formas de iniciar una conferencia:

- La conferencia puede iniciarse en un tiempo preestablecido por el servicio, en el tiempo de comienzo indicado en la reserva. La aplicación es notificada de esto. Entonces, puede añadir participantes a la conferencia o los participantes pueden agregarse a la conferencia marcando una dirección (número de teléfono) suministrado durante la reserva.
- La conferencia puede ser creada directamente a petición de la aplicación.

La interfaz de Control de Llamada de Conferencia hereda de la interfaz de Control de Llamada Multimedia y ésta a su vez de la interfaz de Control de Llamada Multi-Participe. Es posible implementar el control de llamada de conferencia sin ninguna funcionalidad multimedia usando sólo los métodos heredados de MPCC además de los métodos propios.

El servicio consta de 2 paquetes, uno para las clases del lado de la aplicación y otro para las del lado del servicio. En las siguientes figuras se muestran los diagramas de clases.

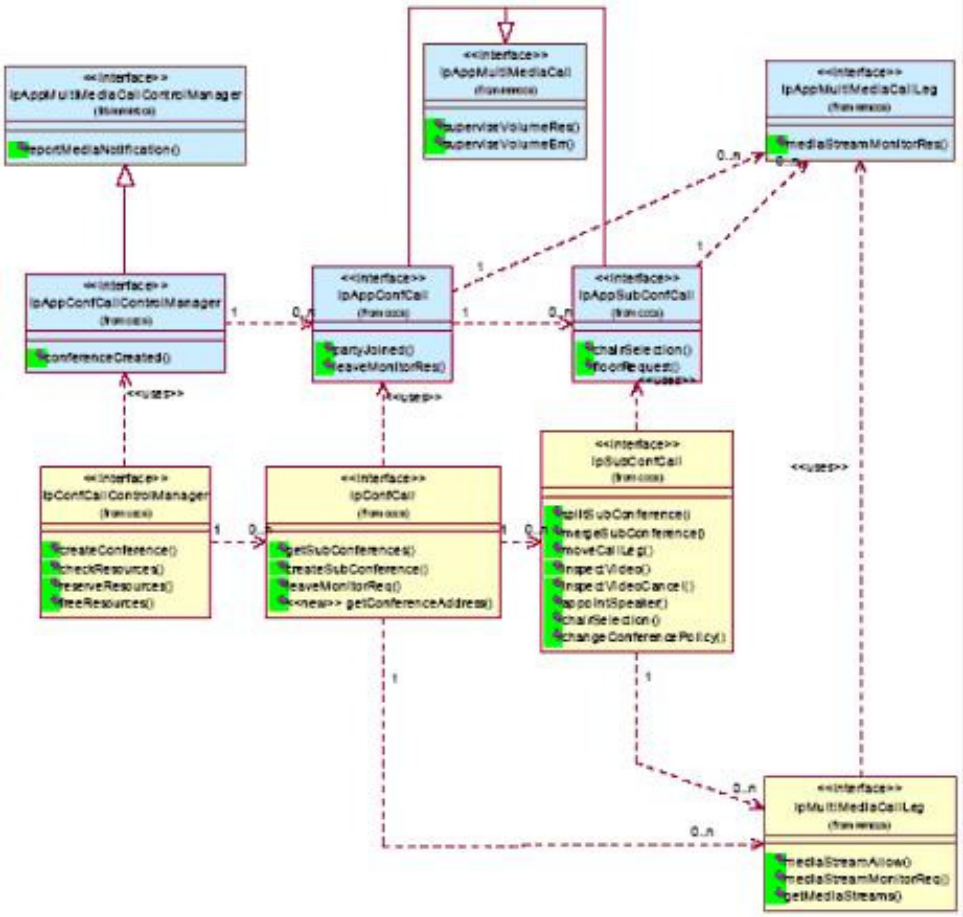

Figura 10. ***CCCS. Clases de la aplicación***

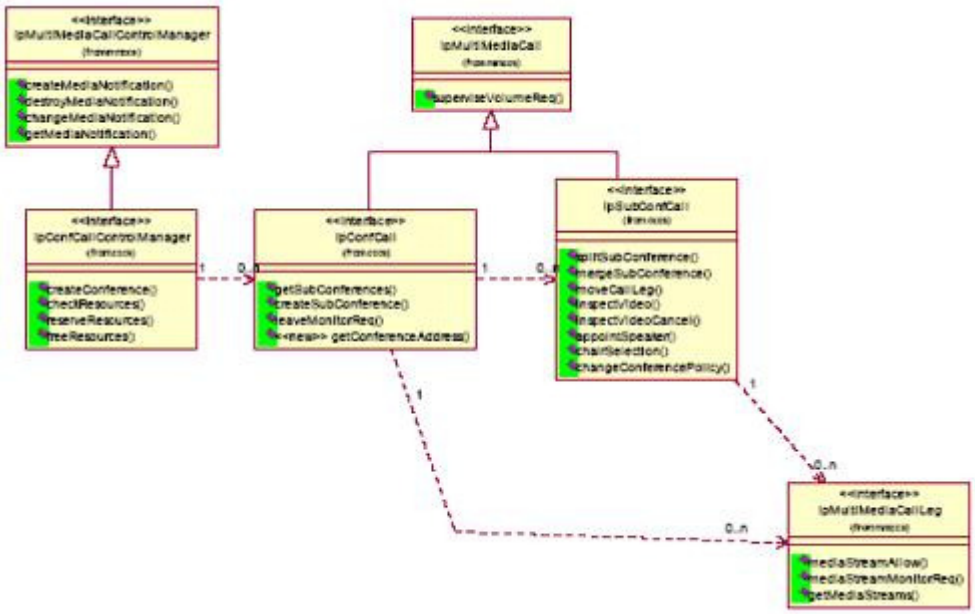

Figura 11. ***CCCS. Clases del servicio***

2.7.2 SCF Interacción de Usuario

El servicio Interacción de Usuario (UI – *User Interaction*) permite a las aplicaciones interactuar con los usuarios finales. Consta de tres interfaces:

- Gestor de Interacción de usuario: Contiene funciones de gestión de asuntos relacionados con la interacción con el usuario.
- Interacción de Usuario Genérica (GUI – *Generic User Interaction*). Contiene métodos para interactuar con el usuario final.
- Interacción de Usuario de Llamada (CUI – *Call User Interaction*). Tiene métodos para interactuar con el usuario final cuando éste está realizando una llamada.

GUI proporciona funciones para enviar información al usuario final o para obtenerla de él. Así permite a las aplicaciones enviar mensajes SMS y USSD por ejemplo. Una aplicación puede usar esta interfaz independientemente de otras SCF's. CUI es equivalente a GUI pero mientras el usuario está vinculado a una llamada, es decir, es participante de una llamada. De hecho, CUI hereda de GUI. En la siguiente figura se muestra el diagrama de clases.

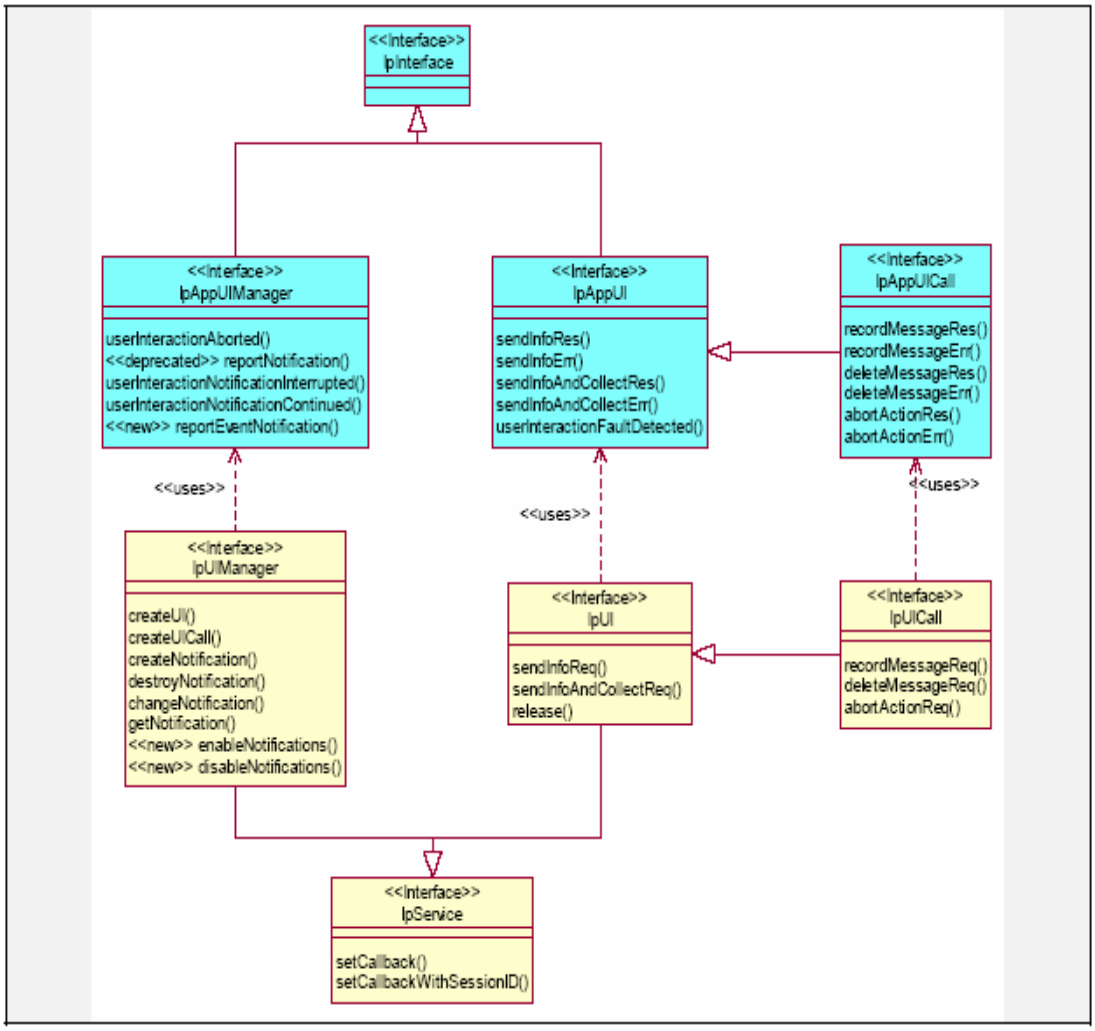

Figura 12. Paquete Interacción de Usuario Genérica

2.7.3 SCF Movilidad

El servicio de Movilidad (*Mobility SCF*) es un conjunto de servicios usados habitualmente para implementar aplicaciones relacionadas estrechamente con la movilidad del usuario. Consta de 4 interfaces o servicios:

- Localización de Usuario (UL – *User Location*). Proporciona un servicio de localización geográfica general. Con este servicio las aplicaciones pueden obtener la localización geográfica y el estado de usuarios de telefonía fija, móvil o basada en IP.
- Localización de Usuario Camel (ULC - *User Location Camel*). Proporciona información de localización basada en la información de la red, en vez de con

coordenadas geográficas usadas en UL. Con las funciones de ULC, una aplicación puede solicitar el número del VLR, la localización de un Identificador de Área y de un Identificador Global de Celda y otros datos de localización.

- Localización de Usuario de Emergencia (ULE – *User Location Emergency*). En el caso de llamadas de emergencia, la red puede localizar al llamante automáticamente. La localización obtenida se envía directamente a la aplicación encargada de manejar la localización de usuario en casos de emergencia.
- Estado del usuario (US – *User Status*). Proporciona un servicio de estado del usuario general, permitiendo conocer cuál es el estado de usuario de telefonía fija, móvil o basada en IP.

En UL y ULC la aplicación puede obtener información de localización de un usuario de las siguientes maneras:

- Cuando se produzca un evento que cumpla determinados criterios, por ejemplo cuando se produzca una variación en la situación del usuario.
- De manera periódica, indicando un intervalo deseado.
- Por petición explícita de la aplicación.

En ULE y US no se puede obtener información de manera periódica, pero sí explícitamente y cuando ocurra un evento. Este evento en ULE es la creación de una llamada de emergencia y en US podría ser el cambio de estado del usuario.

Muchas de las funciones son asíncronas para que la aplicación pueda manejar más información simultáneamente. Las aplicaciones deben realizar las correspondientes interfaces para obtener los resultados e informes.

En la siguiente figura se muestran los diagramas de clases.

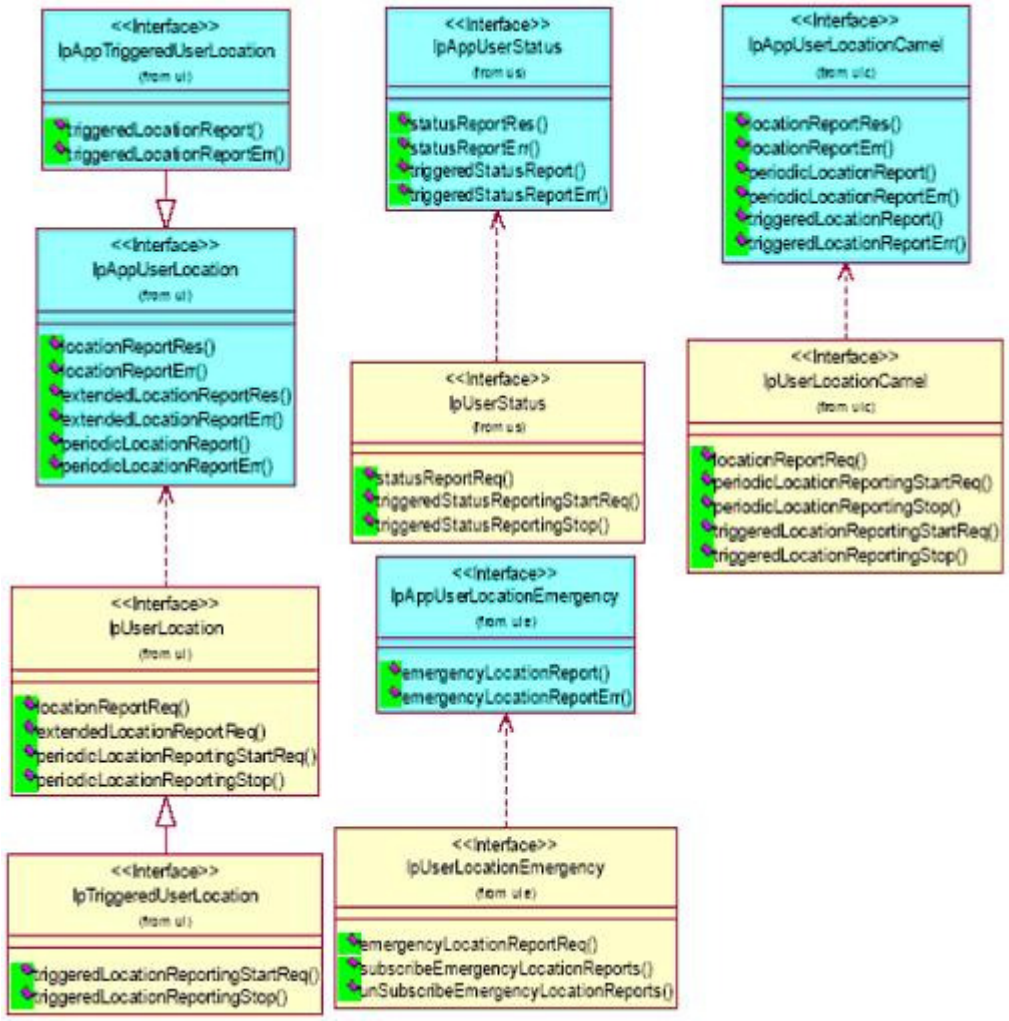

Figura 13. ***Paquete Movilidad***

3. MiLife™ ISG SDK

3.1 Introducción

En la realización del presente proyecto se ha utilizado la herramienta software *MiLife™ ISG SDK* 7.1 de Lucent Technologies. Dicha herramienta se ofrecía de forma gratuita (sólo había que registrarse primero) en la Web de Lucent: *http://www.lucent.com/developer/milife*.

Esto no quiere decir que se pueda emplear este software con fines lucrativos: es una herramienta de aprendizaje y simulación de conceptos cuya puesta en práctica en la vida real es bastante costosa por todos los recursos que necesita.

Hay que señalar que, desde un par de meses después de la adquisición por mi parte de dicho software, la Web de Lucent cambió, pasando a ser hasta ahora *http://www.alcatel-lucent.com*, en la que este software ha dejado de estar disponible.

La aplicación se estructura en dos partes bien diferenciadas: *MiLife™ ISG* y *MiLife™ ISG SDK*. A continuación se hará una pequeña introducción a las mismas.

3.1.1 MiLife™ ISG

MiLife™ ISG (*Intelligent Services Gateway*) provee un camino estándar para que los operadores faciliten sus recursos de red a los proveedores de servicios de aplicaciones (ASPs –*Applications Service Providers*) y a los desarrolladores de aplicaciones cliente 3G. ISG esconde los detalles subyacentes de la red y protege a los desarrolladores de aplicaciones de las complejidades de las redes de telecomunicaciones.

ISG posee las siguientes **características**:

- Un conjunto de métodos OSA (*Open Service Architecture*) [en forma de APIs], que ofrecen un acceso seguro a los recursos internos de la red.
- Las APIs están definidas mediante los estándares de OSA/Parlay y también en IDL (CORBA). Las aplicaciones cliente creadas por los desarrolladores serán capaces de enviar peticiones a la ISG a través de estas APIs.
- Un juego de Servidores de Capacidades de Servicio (*SCS – Service Capability Servers*) que son los que proveen la interfaz y funcionalidad para interactuar con los elementos de la red. Estos SCSs no son más que abstracciones de las funcionalidades de la red, permitiendo de esta manera separar los detalles técnicos de la red de la implementación de los servicios.

ISG está basado y soporta los siguientes **estándares**:

- 3GPP OSA Release 6 (ver 3GPP TS 29.198 y *http://www.3gpp.org* para más información)
- Parlay 4.1 (ver punto anterior y/o *http://www.parlay.org* para más información)
- PAM Specification Document – Version 1.0 (ver *http://www.pamforum.org* para más información)
- CORBA 2.3 (ver *http://www.corba.org* para más información).

Para entender de una manera adecuada lo que nos ofrece *MiLife™ ISG* es aconsejable recordar algo que ya se comentó en el punto anterior: la arquitectura típica de OSA/Parlay. Para ello, se vuelve a mostrar a continuación la figura relacionada con dicha arquitectura:

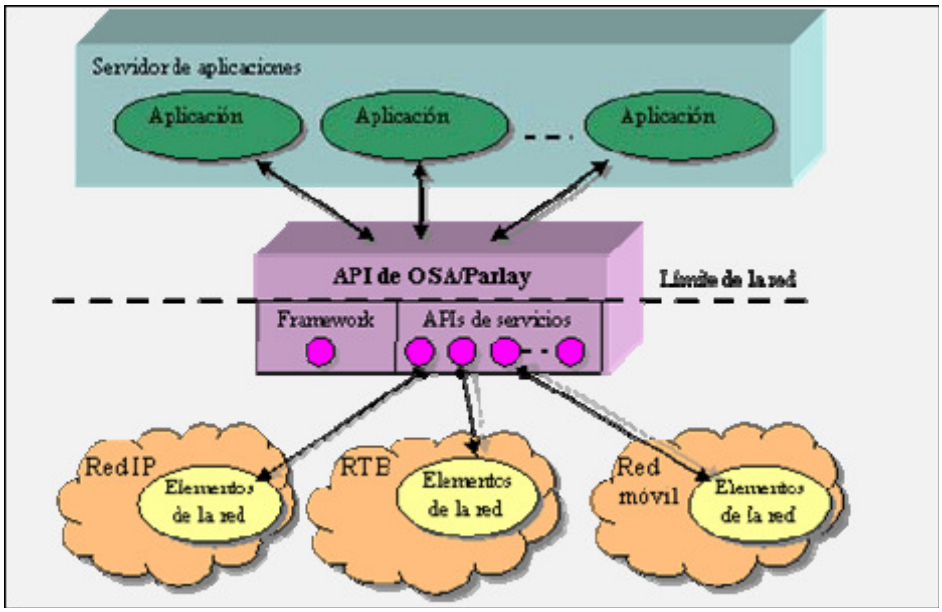

Figura 14. ***Arquitectura típica de OSA/Parlay***

El bloque central que se puede apreciar en la figura correspondería a la pasarela o *gateway* necesaria en este tipo de sistemas, y a través de la cual se canalizan todas las interacciones OSA/Parlay hacia la red y viceversa. Pues bien, *MiLife™ ISG* hace las funciones de esa pasarela o plataforma OSA/Parlay. Como se aprecia en la figura, es en este lugar donde reside el *Framework* o Armazón, y será este dispositivo el que se preocupe de cómo y donde obtener la información de la red que se solicite desde nuestras aplicaciones. Como se verá más adelante, una de las cosas más útiles de utilizar este software, es que nos ofrece una API más sencilla de manejar que la de OSA/Parlay.

Podemos ver como las aplicaciones clientes residen en un servidor de aplicaciones y acceden a los Servidores de Capacidades de Servicio, SCSs, a través de la API de OSA/Parlay del ISG. Estas APIs constituyen el adhesivo entre dichas aplicaciones y los SCSs. De esta manera, las aplicaciones pueden llegar a ser independientes de la tecnología de la red que haya por debajo.

Los SCSs proveen a las aplicaciones clientes los SCFs (Funcionalidades de Capacidades de Servicio) de OSA. Un SCS puede tener múltiples SCFs. Los recursos de la red son los elementos de la misma, así como los protocolos usados.

3.1.2 MiLife™ ISG SDK

MiLife™ ISG SDK (*Software Development Kit*) es una colección de herramientas, documentación y código que facilita el desarrollo y el testeo de aplicaciones basadas en el ISG de Lucent. Como cualquier sistema de desarrollo software, el SDK contiene librerías, ejemplos y documentación necesarios para crear aplicaciones propias.

Pero MiLife™ ISG SDK, además, incluye un potente simulador, que permite probar las aplicaciones creadas sin la necesidad de recurrir a otros sistemas de prueba, más reales, pero a la vez mucho más costosos.

El diagrama que se aprecia a continuación muestra la estructura básica de MiLife™ ISG SDK:

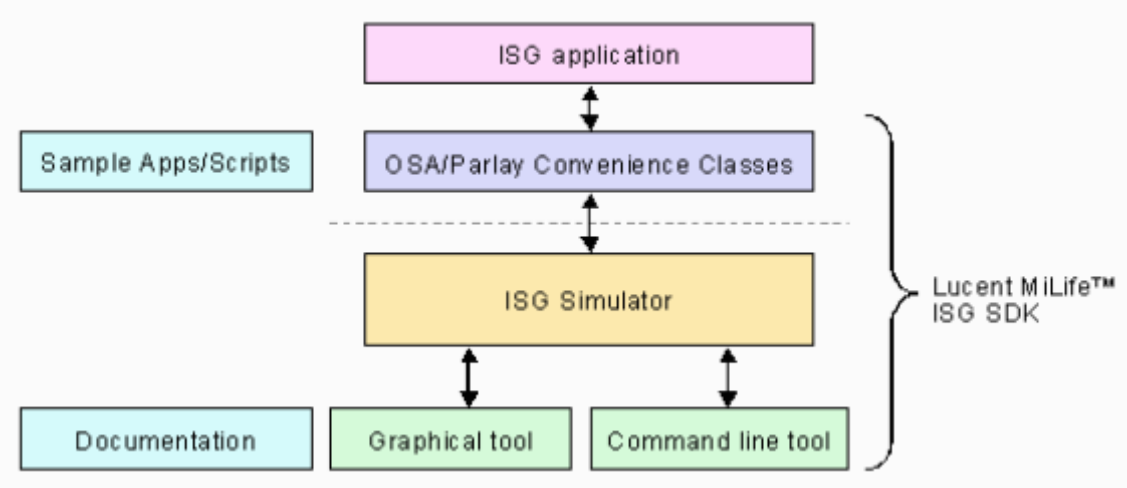

Figura 15. ***Estructura de MiLife™ ISG SDK***

Como aparece en la figura, la estructura básica consiste en los siguientes componentes:

- ***Convenience Classes***

  De la misma manera que MiLife™ ISG, con el SDK podemos utilizar directamente las APIs de OSA/Parlay (en CORBA), y en diversos ejemplos

suministrados podemos ver el uso de las mismas. Pero a la vez se comprueba que el uso de dichas interfaces no es trivial. Es por ello, que este entorno de desarrollo nos ofrece las *Convenience Classes*, que son interfaces Java que esconden la mayor parte de la complejidad de las APIs de OSA/Parlay. Son un nivel de abstracción superior, es decir, las APIs que ofrece no son directamente las de OSA/Parlay, sino que encapsula éstas ofreciendo otras mucho más sencillas de utilizar, en Java, y que ahorra enormemente el proceso de creación de aplicaciones. Evidentemente, esta abstracción provoca también una simplificación en muchas de las funcionalidades, pero suficientes para los objetivos que se persiguen en el presente proyecto.

- ***Simulador y herramientas***

  El simulador ISG proporciona toda la funcionalidad necesaria para que una aplicación recupere, reciba e intercambie datos a través de las interfaces de OSA/Parlay. El simulador no está conectado con equipos de una red real. Se puede controlar a través de una herramienta gráfica o a través de la línea de comandos. La herramienta gráfica, que es la que se ha usado en la realización de este proyecto para la simulación, permite, entre otras cosas, ejecutar scripts, así como cambiar y ver el estado del simulador.

- ***Documentación***

  Como se ha comentado más arriba, el software viene acompañado de guías de referencia, tutoriales y del manual del usuario para el simulador. Además incluye todas las librerías necesarias para poder trabajar con otros entornos de desarrollo, como por ejemplo Eclipse, que es el que se ha usado en la realización de este proyecto.

- ***Aplicaciones de ejemplo y scripts***

  Como ayuda para empezar, el SDK incluye algunos ejemplos de aplicaciones con las funcionalidades más significativas. Para ejecutar las aplicaciones primero hay que hacer correr el simulador. Éste hace uso de un script, escrito en XML, que le da toda la información que necesita para un comportamiento

dinámico durante la simulación. Por ejemplo, en el script se puede definir que un abonado camine desde un punto A hasta otro B con una cierta velocidad. El SDK también contiene scripts de ejemplo.

3.2 Simulador ISG

Este punto describe más en detalle las características y el uso del simulador de MiLife™ ISG SDK. Nos vamos a centrar sobre todo en la herramienta gráfica del mismo, pues ha sido la más usada. También se describirá la estructura de los archivos XML usados para crear el entorno de pruebas de la aplicación.

En primer lugar profundizaremos en la estructura del paquete SDK y en algunas de sus principales características. A continuación se describirán los pasos más importantes que se han seguido para configurar y usar tanto el simulador como la herramienta gráfica, describiendo los paneles de simulación, elementos que forman parte de la herramienta gráfica y que permiten llevar a cabo la simulación.

3.2.1 Estructura del SDK

A continuación se muestra una figura que muestra la estructura del SDK con un nivel de detalle mucho mayor que el mostrado en el punto anterior:

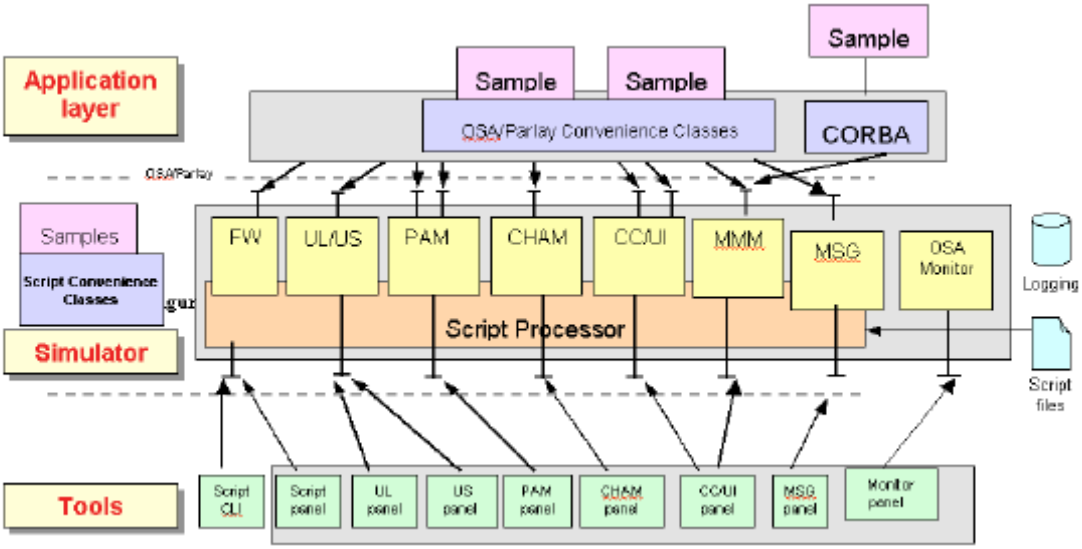

Figura 16. ***Estructura detallada del SDK***

En la figura podemos apreciar que existen tres capas bien diferenciadas: la de aplicación, la del simulador y la de herramientas. En la capa de aplicación es donde residirán las aplicaciones creadas, que podrán usar la API de OSA/Parlay directamente o bien las OSA/Parlay *Convenience Classes* suministradas por el SDK. Estas aplicaciones se comunicarán con el simulador a través de la interfaz OSA/Parlay, y éste hará uso de las herramientas suministradas por la herramienta gráfica para poder suministrar a la aplicación toda información que solicite acerca de la red.

El simulador SDK es, en realidad, una colección de simuladores, uno por cada uno de los siguientes SCSs (Service Capability Servers):

- **Framework (FW):** simula el armazón, pieza necesaria en este tipo de arquitectura de servicios, y con la que es necesaria establecer la comunicación inicial para poder obtener el resto de funcionalidades.
- **User Location (UL):** ofrece información acerca de las posiciones de los terminales siguiendo distintos procedimientos (cuando se solicite, periódicamente o por eventos previamente especificados).
- **User Status (US):** ofrece información sobre el estado de los terminales (ocupado, apagado o encendido), y al igual que el anterior, también de diversas maneras (cuando se solicite o por eventos programados).

- **Presence and Availability Management (PAM).**
- **Charging and Accounting Management (CHAM):** permite manejar la tarificación de los abonados.
- **Call Control (CC):** ofrece todas las funcionalidades relacionadas con una llamada telefónica.
- **User Interaction (UI):** simulador que ofrece la capacidad de reproducir mensajes predefinidos a través de una llamada y poder obtener respuesta del oyente (a través de la pulsación o pulsaciones de las teclas del terminal).
- **Messaging (MSG):** ofrece la capacidad de envío y detección de llegada de correos electrónicos.
- **Multi-Media Messaging (MMM):** ofrece la capacidad de envío y detección de llegada de mensajes (MMS).

El simulador incluye un monitor de mensajes OSA/Parlay entre la aplicación de usuario y él mismo. Además, los eventos más importantes ocurridos durante la simulación se copian en un fichero (log). El bloque *Script Processor* es usado para cargar y ejecutar scripts en la herramienta gráfica desde la aplicación.

El simulador está inicialmente vacío, lo que significa que no contiene ningún tipo de información de simulación, como son los abonados y sus comportamientos. La única manera de llenarlo de información es a través de los scripts. Aunque se pueden hacer cambios del estado del simulador con la herramienta gráfica, como por ejemplo cambiar la localización o estado de un abonado, sus posibilidades son limitadas. Los scripts ofrecen un mayor control para la simulación, desde el punto de vista de la precisión y de la temporalidad.

Todos los SCSs simulados soportan el concepto de abonado. Un abonado se identifica mediante su MSISDN, que es un número de 12 dígitos. El nombre del abonado, definido en el script, es también visible en la herramienta gráfica. Como el nombre del abonado puede ser elegido libremente, es más fácil referenciar un nombre

que un número (que es lo que es el MSISDN). El simulador ISG también soporta el concepto de grupos, que son tratados por la herramienta gráfica como un único abonado (desde el punto de vista de la localización, estado,...) con un rango de distintos MSISDNs. Esto es útil cuando hay que realizar pruebas con multitud de abonados. El concepto de grupo no es soportado por la interfaz OSA/Parlay, sin embargo una petición de cualquier abonado que es parte de un grupo es una solicitud válida, de la misma manera que si un grupo hace una llamada en el simulador, las aplicaciones pueden recibir disparos por cada uno de los abonados de ese grupo.

Como ya se ha comentado anteriormente, el simulador ofrece una serie de herramientas para trabajar, una gráfica, y otra desde línea de comandos.

- **Herramienta gráfica**. El simulador se puede arrancar desde dentro de la herramienta gráfica. Ésta provee una manera para ver y controlar los scripts, los SCSs y el monitor de mensajes OSA/Parlay CORBA. Esta herramienta consiste en un conjunto de herramientas que corren bajo la misma aplicación. Puede estar desconectada o conectada con el simulador. En éste último caso, y con un script cargado, la apariencia de la misma es más o menos como se puede apreciar en la siguiente figura:

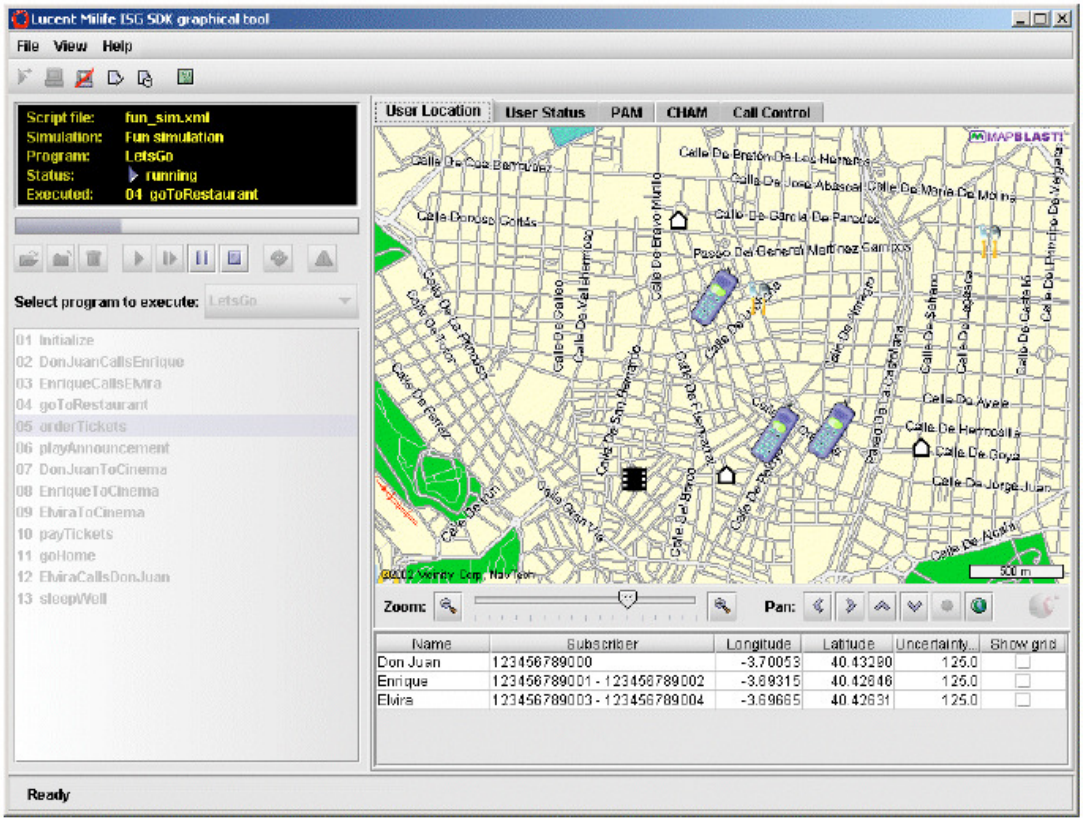

Figura 17. ***Herramienta gráfica del simulador ISG***

- **Herramienta de línea de comandos**. Permite controlar el simulador mediante scripts. Con esta herramienta el usuario puede comprobar el estado del *Script Processor* y cargar y ejecutar scripts. Normalmente se utiliza para procesamiento de archivos por lotes.

3.2.2 Configuración del simulador ISG

Gracias a la comunicación CORBA entre varios elementos, las aplicaciones cliente, el simulador y las herramientas (tanto gráfica como la de línea de comandos) pueden correr en diferentes ordenadores. De la misma manera, la herramienta gráfica y el simulador pueden ejecutarse bajo la misma Máquina virtual de Java (JVM), a diferencia de la herramienta de línea de comandos y aplicaciones que no pueden correr en la misma JVM que el simulador. Por esta razón, ha sido necesario instalar dos máquinas virtuales en el ordenador en el que se ha desarrollado el proyecto, puesto que el simulador y las aplicaciones corrían en la misma máquina. Como el simulador, según

el fabricante, estaba probado para JSE v 1.4, es esta JVM la que soporta el simulador. A parte, se instaló el JDK 6, que es el que se usa para hacer correr las aplicaciones.

Con la herramienta gráfica de Lucent MiLife™ ISG SDK podemos realizar lo siguiente:

- Arrancar el simulador: es la manera más fácil de hacerlo. Se puede configurar la herramienta gráfica (y es la configuración utilizada en el presente proyecto) para que una vez arrancada establezca conexión con el simulador, corra éste en la misma máquina o en otra diferente. Esto se hace a través del fichero *isgsimgui.properties* o también con una opción de la herramienta gráfica. Una vez haya arrancado la aplicación, la apariencia del mismo es la misma que la mostrada en la figura anterior, pero sin información alguna cargada (el panel de los scripts, que es la parte de la izquierda aparecerá vacía, así como todo lo relacionado con los abonados).

- Conectar a un simulador remoto: aunque se puede hacer de manera automática, si el simulador está en una máquina remota es preferible hacerlo de manera manual. El simulador soporta que se conecten múltiples clientes, ya sea desde la herramienta gráfica o desde la de línea de comandos. La conexión desde la herramienta gráfica, que es la que se ha utilizado, se ilustra con la figura 19:

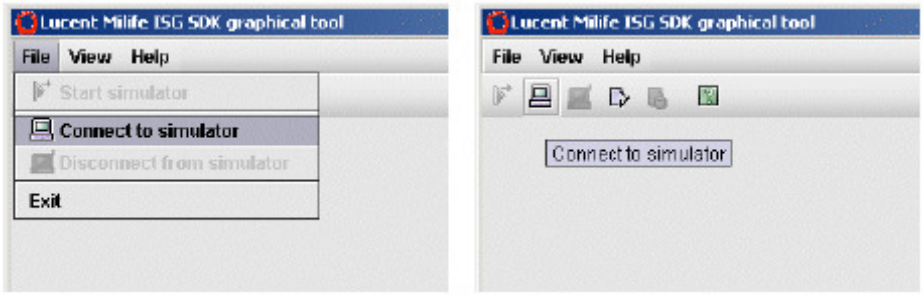

Figura 18. ***Conexión con el simulador***

Una vez establecida la conexión, se puede cargar un script (mediante el panel de script) y ejecutarlo, para comenzar así la simulación.

- Desconectar el simulador: como se puede apreciar en la figura anterior, en el menú File tenemos la opción de desconexión, o simplemente podemos cerrar la aplicación.

La configuración del SDK, y por ello del simulador y la herramienta gráfica, se puede realizar de las siguientes maneras:

- Desde el menú *View* de la herramienta gráfica.

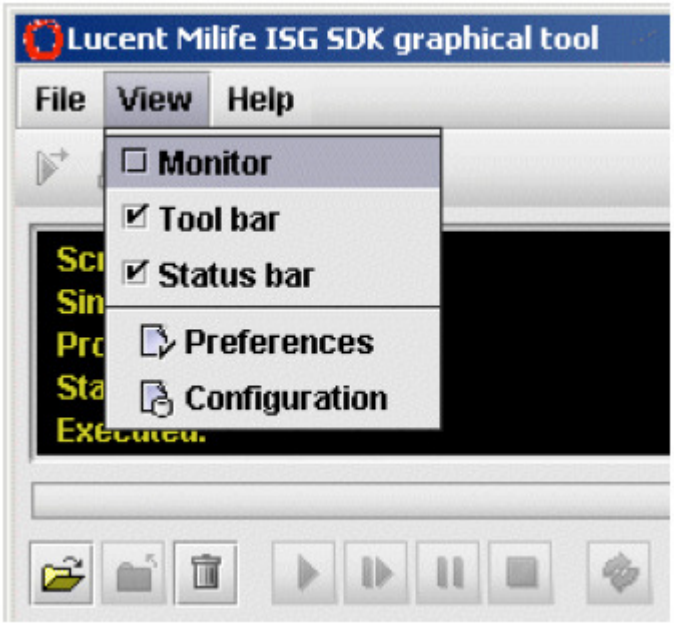

Figura 19. ***Menú View de la herramienta gráfica***

Desde este menú se puede seleccionar activar o desactivar ciertas partes de la herramienta y nos permite acceder a los submenús *Preferences* y *Configuration*. A continuación se muestran imágenes de cada submenú con las configuraciones escogidas.

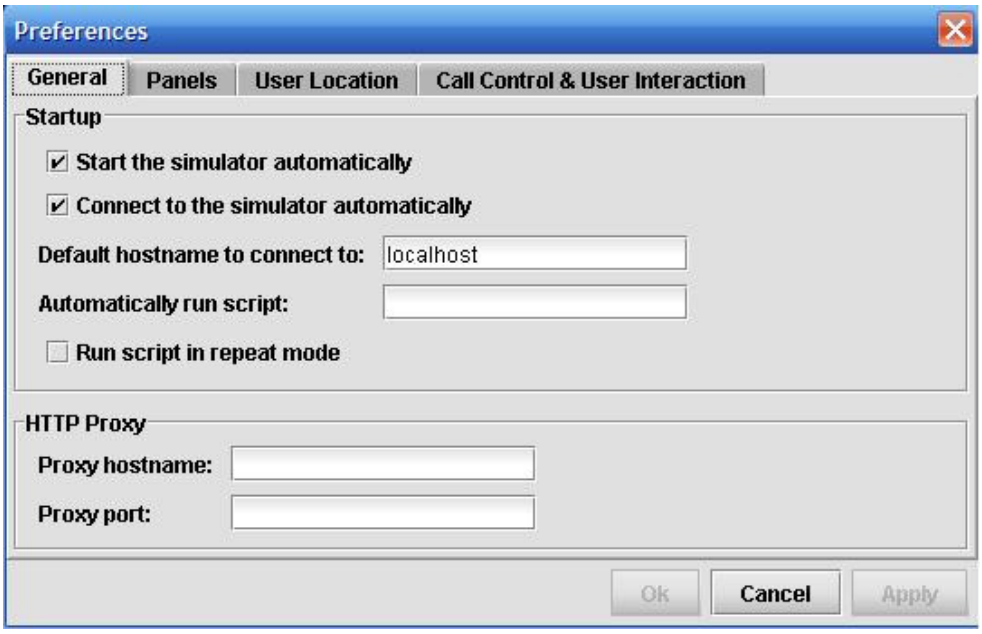

Figura 20. ***Pestaña General del Menú Preferentes***

En esta ventana encontramos varias pestañas. En la primera de ellas, simplemente se ha escogido que la aplicación arranque y se conecte al simulador automáticamente. Como ya se dijo anteriormente, se ha utilizado el mismo ordenador para todo, por lo que el simulador corre en localhost. A continuación se muestran las demás pestañas con las opciones utilizadas.

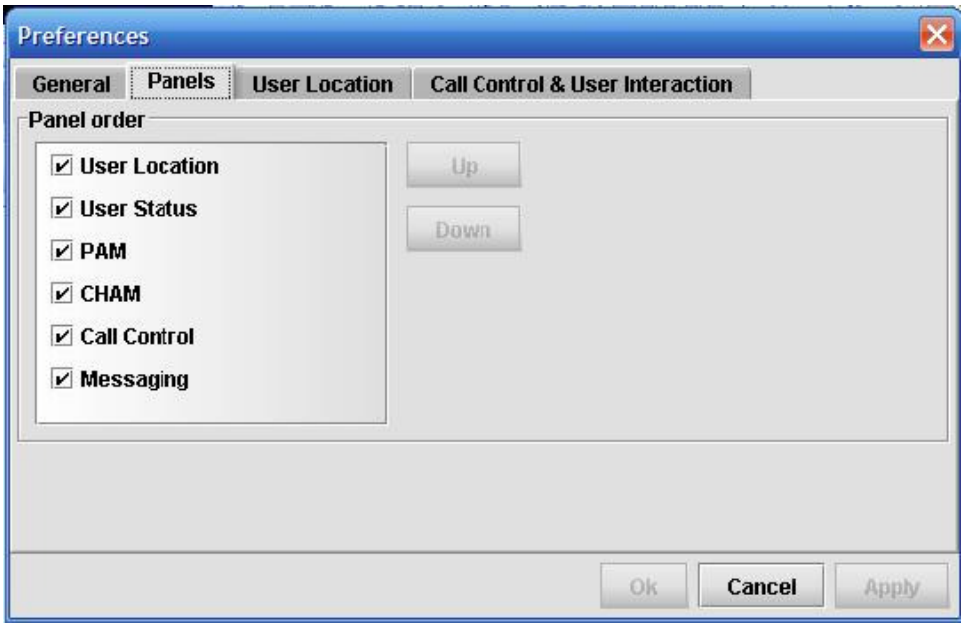

Figura 21. ***Pestaña Panels del Menú Preferences***

La herramienta gráfica se compone de distintos paneles que engloban a todos los SCSc. En esta pantallas podemos escoger los paneles, y por tanto los SCSs que queremos que aparezcan, pues a lo mejor nuestra aplicación no hace uso de ellos.

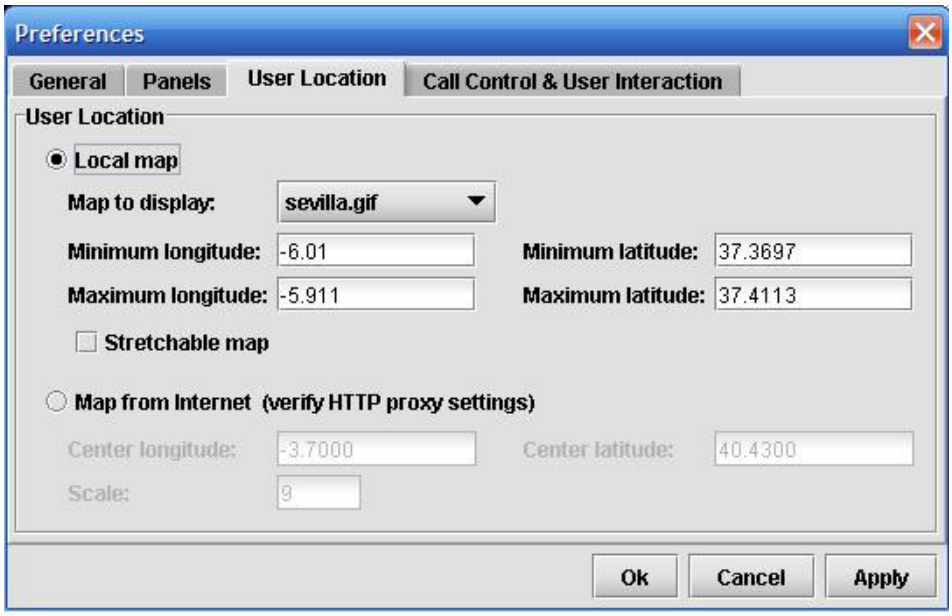

Figura 22. Pestaña User Location del Menú Preferences

En esta pantalla se pueden configurar algunas opciones del SCS User Location. Se puede escoger entre un mapa almacenado localmente o uno escogido de Internet.

A continuación se muestra la última pestaña de esta pantalla, que contiene algunas características de aspecto y uso del panel *Call Control & MMM*, que es en el que aparecen los terminales de los abonados cargados en el simulador. La configuración de *User Interaction* también se incluye aquí, pues está íntimamente relacionada con las llamadas.

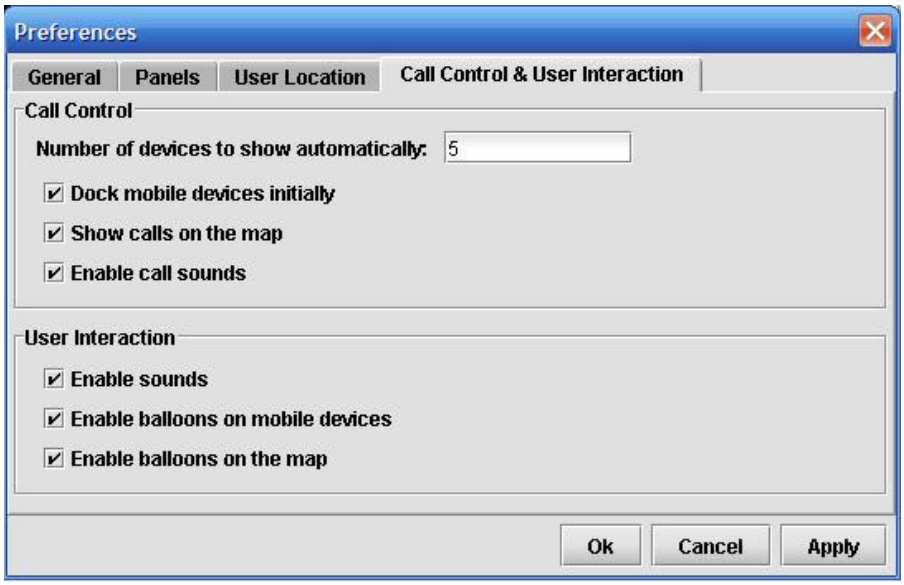

Figura 23. Pestaña Call Control & User Interaction del Menú Preferences

A continuación se muestran las otras opciones elegidas del submenú *Configuration,* que al igual que el *Preferences*, es una pantalla con diversas pestañas.

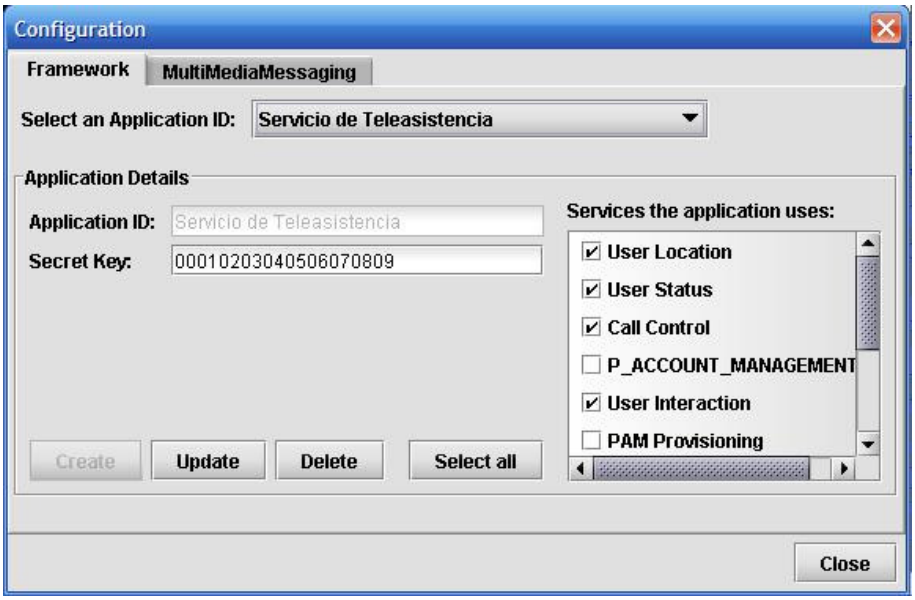

Figura 24. ***Pestaña Framework del Menú Configuration***

Cada aplicación en la base de datos del Framework tiene asociada un identificador de aplicación (Application ID), usado para autentificar la aplicación en el simulador. Unido a cada identificador de aplicación están los servicios que la aplicación puede usar (Servidores de Capacidades de Servicio, SCSs). En esta pantalla podemos ver como podemos borrar, actualizar o crear nuevos identificadores de aplicación, crear su contraseña (*Secret Key*) y asociarle a cada uno de ellos los servicios que vayan a usar. Se puede ver en la imagen el creado para este proyecto. Esta tarea se puede hacer también a través de los scripts, de manera que cada vez que se ejecutan, crean el *Aplication ID* si no estaba creado.

Figura 25. Pestaña MultiMediaMessaging del Menú Configuration

Esta pestaña, MultiMediaMessaging, puede ser usada para activar o desactivar algunos escenarios de error en el envío de mensajes multimedia, para hacer la simulación más realista. Entre ellos podemos destacar:

- Activar o desactivar las respuestas en caso de fallo.
- Incluir texto extra para que sea lanzado a través de la excepción para los fallos de envío y cancelación de mensajes.
- Incluir un número de abonado al que enviar el mensaje de entrega fallida.

Estas configuraciones permiten al usuario activar errores para así comprobar las distintas excepciones. Estas excepciones se producen en la red, pero se configuran en el simulador pues es éste el que simula el comportamiento de la misma.

- Editando las propiedades del simulador.

Se ha visto cómo se pueden configurar ciertas características de la herramienta gráfica a través de los menús y barras de herramientas de la misma, pero la mayoría de características, sobre todo del panel de User Location, que es el que

necesita de una mayor definición de características para crear el entorno deseado (lugares del mapa, imágenes de lugares, posicionamiento de dichos lugares,…), se tienen que hacer a través del fichero de configuración **isgsimgui.properties**.

En cuanto a lo que el simulador se refiere, desde la herramienta gráfica solamente se puede configurar el identificador de aplicación para el Framework. Las propiedades específicas del simulador están definidas en el archivo **isgsimulator.properties**. En este fichero de pueden modificar multitud de características de la red, para cada uno de los SCSs por separado. A modo de ejemplo, se muestran un par de líneas de dicho archivo:

- *isgsim.us.network_type=UMTS*

  Establece el tipo de red para el SCS User Status. Cuando su valor es UMTS, el disparo de eventos por parte de la red será soportado por el simulador. Cuando el valor es CDMA, no.

- *isgsim.ul.network_type=CDMA*

  Hace lo mismo que la línea anterior pero para el simulador User Location.

3.2.3 Paneles de simulación

Ya se ha descrito, sin entrar en demasiados detalles, la configuración básica que se ha utilizado en el simulador y en la herramienta gráfica de MiLife™ ISG SDK. Ahora se van a describir brevemente los paneles de que consta dicha herramienta, y que son los que permiten interactuar con las aplicaciones creadas, para tratar de conseguir un realismo lo más alto posible. Nos centraremos en los que se han usado en el desarrollo de la aplicación de teleasistencia.

Si volvemos a mirar la figura 17 (Estructura detallada del SDK) veremos como la herramienta gráfica se compone de una serie de paneles, y son éstos los que intercambian directamente información son los SCSs correspondientes del simulador. Cada SCS tiene su propio panel, excepto el de User Interaction, puesto que un abonado de User Interaction no existe hasta que una llamada esté activa. Los anuncios de este SCSs se muestran tanto en el panel User Location y en el Call Control.

Los paneles se pueden ejecutar solos, sin necesidad de arrancar la herramienta gráfica, con lo que sólo se estaría ejecutando el SCS correspondiente en el simulador. Para ello hay que ejecutar "panel.bat" en el directorio bin de la instalación del SDK. Se necesita también pasar el tipo de panel como primer parámetro el archivo de lotes (los tipos son: UL, US, PAM, CHAM, CC, Messaging, Script y Monitor).

El primer panel que se va a mostrar es el de script. Aparece a la izquierda de la herramienta gráfica y contiene información acerca del script cargado en el simulador. En la figura se puede apreciar con un rectángulo de color rojo que lo rodea.

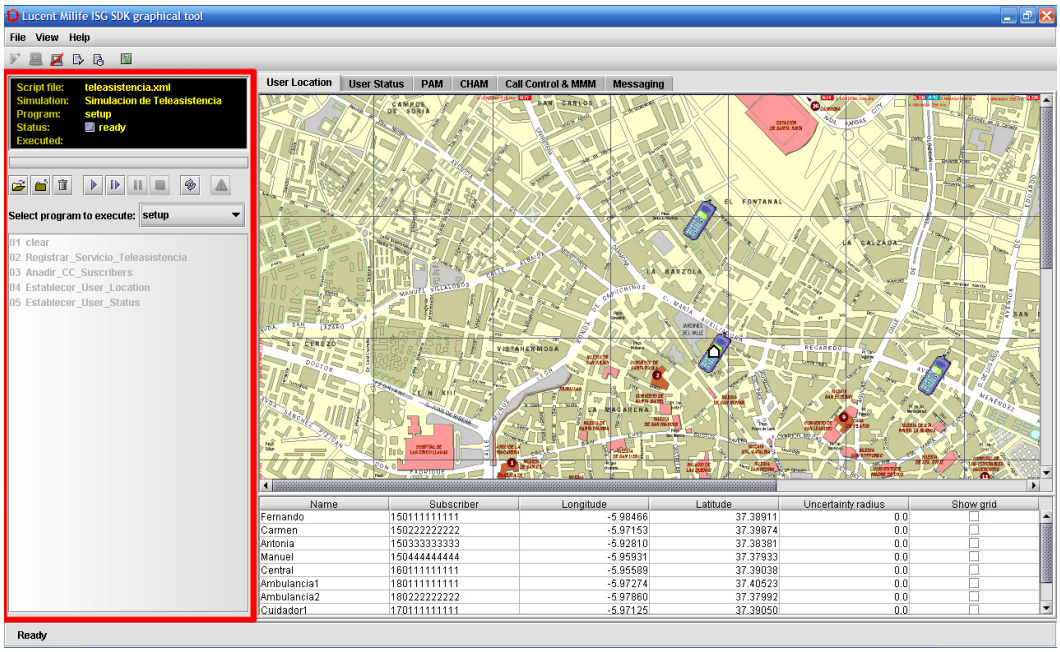

Figura 26. ***Script y User Location Panels***

Podemos realizar varias acciones con los scripts. Dentro de un script se definen uno o varios programas, y dentro de ellos, uno o más pasos. En este panel, podemos elegir el script que queremos cargar, elegir dentro de él el programa que queremos y luego ejecutar los pasos que necesitemos. Se pueden elegir distintos modos de ejecución de scripts (modo continuo o no).

En la misma figura anterior podemos observar también el User Location Panel. Se observa como este panel tiene dos partes bien diferenciadas: una, la de arriba, en la que se muestra el mapa donde se está realizando la simulación, y otra consistente en una tabla que contiene todos los abonados que se encuentran registrados en el SCS User Location. En el mapa, un dispositivo móvil representa a un abonado o grupo de abonados. El movimiento de dicho dispositivo se corresponde con el movimiento del abonado. Además, las llamadas y los anuncios entre abonados se muestran en el mapa para que sea más intuitivo. En la tabla, además del nombre de cada abonado, su número o MSISDN y de su posición en el mapa (longitud y latitud) se puede obtener el radio de incertidumbre.

El simulador User Location soporta dos modelos básicos de red. Se deberá definir en el script cuál se está usando. El primero de ellos es el ideal, en el que el simulador devuelve exactamente la localización del abonado a la aplicación que lo solicite (a través de la interfaz OSA/Parlay). El segundo de ellos es un modelo de rejilla, en el que el simulador devuelve la posición central de la rejilla o celda donde se encuentra el abonado. Si se pica en la opción *Show Grid* de un abonado en cuestión se mostrará en el mapa la celda para ese abonado. El tamaño de la celda en metros es igual al radio de incertidumbre.

En el mapa se pueden añadir imágenes propias para abonados y lugares de interés. Para ello deben añadirse los archivos GIF o JPG en el directorio **etc\images**. Como también se ha comentado antes, se pueden añadir mapas, como ha sido este caso con el de Sevilla. Para ello hay que añadirlos en **etc\maps**. Si se añaden mapas, es también recomendable actualizar el fichero **mapcoordinates.properties** en el directorio anterior. Este fichero contiene las coordenadas de los límites de todos los mapas que se almacenan localmente para el simulador.

A continuación se presenta la imagen del panel User Status. Éste consiste en una tabla con todos los abonados registrados en el SCS correspondiente. En dicha tabla, además del nombre del abonado y su MSISDN respectivo, se muestra el tipo de terminal que posee (*MOBILE, IP* o *FIXED*) y da información de en qué estado se encuentra cada terminal (*REACHABLE*, *NOT REACHABLE* o *BUSY*). Además permite seleccionar

tanto el tipo de terminal como el estado en que queremos que esté cada uno de ellos. Esta característica es la que se ha utilizado en nuestro servicio de teleasistencia. La aplicación detecta cuando algún terminal de los pacientes alcanza un estado de *NOT REACHABLE,* puesto que se ha solicitado previamente a la interfaz OSA/Parlay que notifique dicho evento. Para simular esto, es necesario este panel.

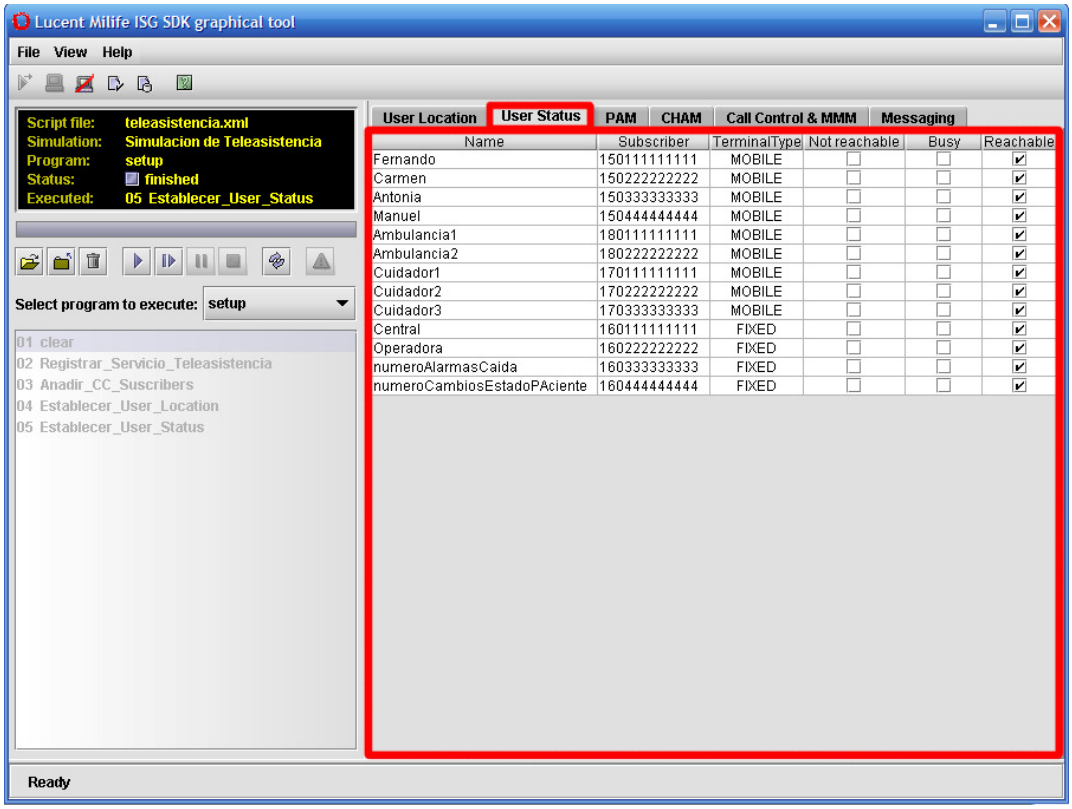

Figura 27. ***User Status Panel***

Por último se va a mostrar el panel relacionado con las llamadas y la mensajería multimedia. Estas funcionalidades comparten panel de simulación. Éste está formado también por dos partes: una serie de terminales con aspecto de teléfono, que representan los dispositivos de los abonados, y una tabla con el estado actual de cada abonado (en lo que a llamadas se refiere). Esto se ilustra con la figura 28.

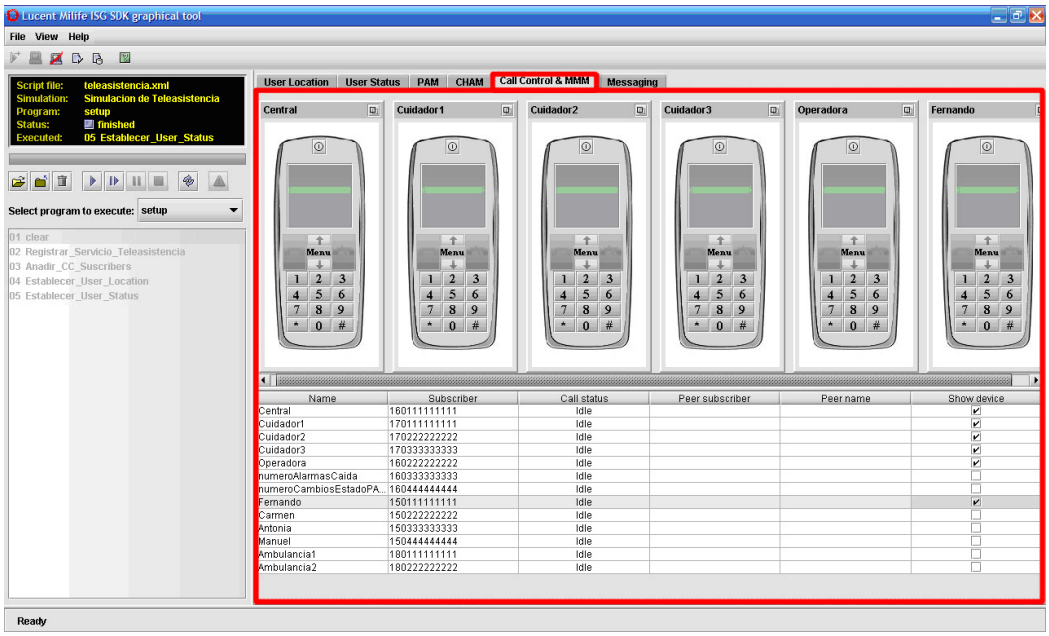

Figura 28. ***Call Control & MMM Panel***

Todos los teléfonos de todos los abonados definidos en el SCS de Call Control no aparecen al arrancar el panel (sólo aparecen un número fijo definido en la configuración de dicho panel). Si queremos ver algún terminal en concreto que no aparece, se activa la casilla *Show device* del abonado en cuestión.

Con estos teléfonos podemos llamar a cualquier abonado definido en el simulador (cuando se establezca una llamada aparecerá en la tabla información relacionada con la misma -número origen, destino,...). Estos dispositivos nos permiten también mandar MMS, accediendo mediante el botón Menú que tienen en el centro y navegando por las opciones que aparecen en la pantalla.

El panel Call Control puede reproducir anuncios correspondientes a la funcionalidad User Interaction y mostrar los textos correspondientes a dichos anuncios encerrados en bocadillos, como puede apreciarse en la siguiente figura:

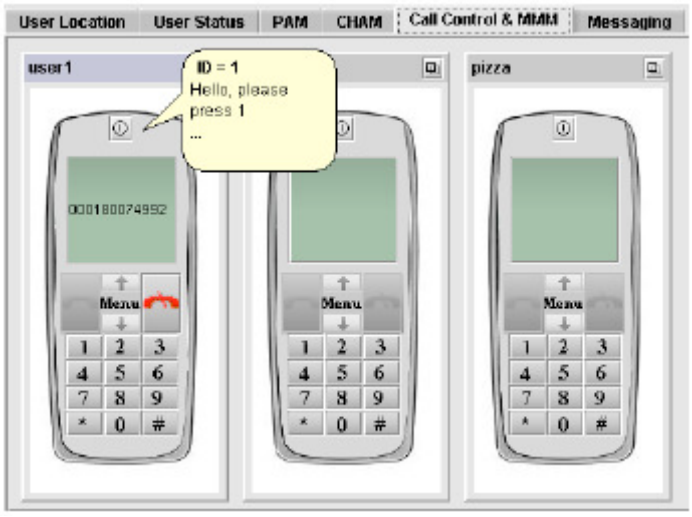

Figura 29. ***Reproducción de mensaje de User Interaction***

Aunque el SCF User Interaction no soporta anuncios de texto, el simulador ISG, los script y la herramienta gráfica sí que lo soportan (mostrándolos a través de bocadillos, y no dentro de los terminales) pero solamente para demostraciones. Por otro lado, el simulador sólo permite que haya interacción con el usuario que inicia la llamada, y no permite recoger información del que la recibe. Esto ha supuesto un problema durante la realización del presente proyecto y se explicará más adelante con detalle, tanto el problema como la solución adoptada.

Destacar también que existen más paneles, correspondientes al resto de SCSs, y cuya apariencia es muy parecida a los ya expuestos. No se entra en más detalles puesto que apenas se han utilizado en la realización del proyecto. Concretamente, además de los explicados, el simulador dispone de los siguientes:

- Messaging
- Presence and Availability Management
- Charging and Accounting
- Monitoring

3.3 Convenience Classes

El objetivo de MiLife™ SDK es facilitar el desarrollo rápido de aplicaciones que tienen como objetivo comunicarse con MiLife™ ISG. Éste soporta la versión R4 de de la API de OSA definida por el *Third Generation Partnership Project* (3GPP) y la versión 3.2 de la API de Parlay definida por el Parlay Forum. Todas estas APIs están definidas en OMG IDL, permitiendo una comunicación directa con el simulador ISG usando CORBA. Alternativamente, el SDK ofrece un conjunto de clases Java que facilitan enormemente el desarrollo de aplicaciones. Estas clases son las *Convenience classes*, y son las que se han utilizado en el desarrollo del servicio de teleasistencia.

Estas clases se pueden dividir en dos partes:

- **OSA/Parlay convenience classes**: proporcionan una API Java que se corresponde con una abstracción de alto nivel de las interfaces OSA/Parlay. Esta API Java ofrece la mayoría de funcionalidades de OSA/Parlay, escondiendo muchas de las partes más engorrosas de la misma.
- **Script convenience classes**: proveen una API Java para controlar el simulador desde un programa Java, por ejemplo para cargar scripts de simulación en el simulador y ejecutarlos.

3.3.1 OSA/Parlay convenience classes

Este conjunto de clases consiste en un número de adaptadores, como se muestra en la figura de más abajo. Cada adaptador ofrece una interfaz simplificada hacia la interfaz OSA/Parlay del SCSs correspondiente. Las principales simplificaciones son:

- Las estructuras más complejas de datos están ocultas.
- Los valores por defecto se usan en la mayor medida posible.
- Los métodos asíncronos se han transformado en síncronos donde ha sido posible.

Con los adaptadores, la tarea de crear una pequeña aplicación y probarla se simplifica, pues estos son capaces de comunicarse con el simulador ISG. La figura 31

ilustra un esquema con los adaptadores existentes y su correspondiente mapeo con los SCSs existentes en MiLife™ SDK.

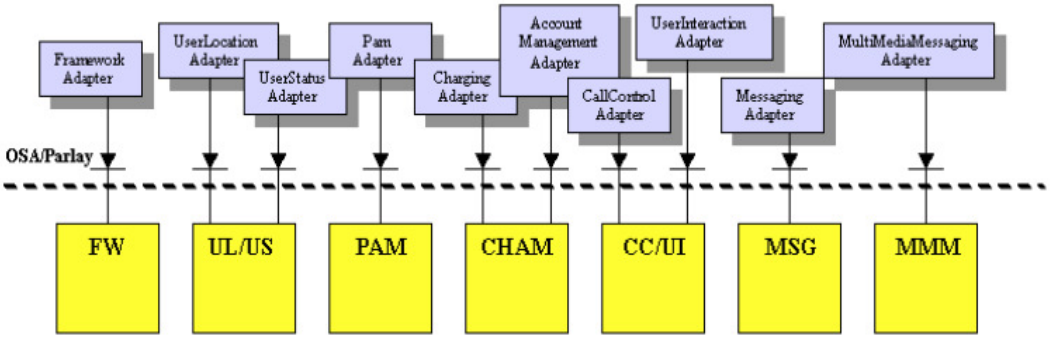

Figura 30. Adaptadores de OSA/Parlay convenience classes

A continuación se van a enumerar los pasos a seguir, y que evidentemente se han seguido en el presente proyecto, para crear una aplicación usando estas clases. Estos pasos solamente se refieren a cómo obtener acceso a los SCSs, ya que una vez que lo tenemos el uso de que hagamos de las funcionalidades de los mismos depende del fin que se quiera obtener con nuestra aplicación.

1. Si usamos el simulador ISG para probar la aplicación, deberemos configurarlo primero para que reconozca la aplicación (suministrándole el identificador de aplicación y el secret key). Como ya se vio en puntos anteriores, esta información se crea bien a partir del script de simulación bien a través de la pestaña de configuración del Framework de la herramienta gráfica. En el caso de una ISG 'real', será el operador el que suministre estas credenciales. En el caso que nos ocupa, esta información de identificación y autentificación la creamos nosotros mismos.

2. En la aplicación, hay que crear un ***FrameworkAdapter*** usando la clase ***FrameworkAdapterFactory***. Se necesita pasarle como parámetro el identificador de aplicación previamente creado y el secret key, de esta manera nuestra aplicación se autentificará con el framework.

3. Mediante el ***FrameworkAdapter*** antes creado, podemos obtener el resto de adaptadores de servicio usando uno de los métodos ***select……Service()***.

4. Usar los adaptadores para acceder a las funcionalidades que necesitemos de cada SCS.

3.3.2 Script convenience classes

Estas clases son mucho menos extensas que las anteriores. Consisten simplemente en dos adaptadores:

- **ScriptAdapter**: ofrece una interfaz simplificada que permite cargar y ejecutar scripts en el simulador.
- **FileMgrAdapter**: ofrece una interfaz con funciones que permiten el manejo de ficheros y carpetas. Este adaptador ofrece una lista de los scripts que están disponibles para el Script Processor (que es una parte del simulador ISG).

Además de estos adaptadores, existe la interfaz ScriptCallback. Una clase que implemente esta interfaz recibirá notificaciones sobre los eventos más importantes relacionados con las ejecuciones de los scripts. Todo esto se ilustra en la siguiente figura:

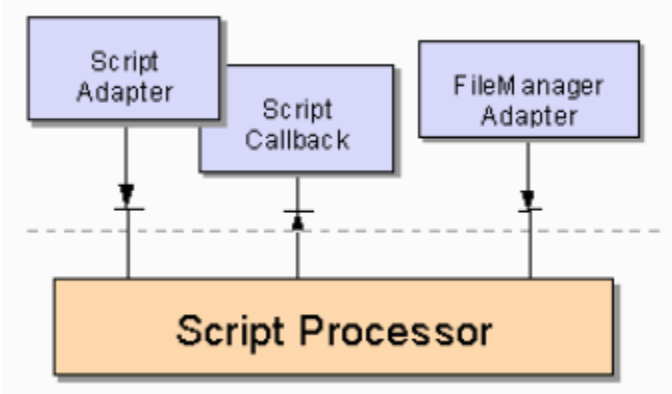

Figura 31. ***Adaptadores de Script convenience classes***

3.4 Scripts XML

En este punto se va a comentar las características principales del formato de los scripts con los que hay que abastecer al simulador para establecer las condiciones

necesarias para las pruebas de simulación. Hay que definirlos en XML. MiLife™ SDK define una serie de esquemas (*schemas*) que hay que seguir para que el simulador, a través del script panel de la herramienta gráfica, reconozca los scripts como válidos.

Todo script de simulación debe contener como elemento raíz la etiqueta **<simulation>**. Esta etiqueta representa una simulación. Los elementos hijo de ésta se muestran en el siguiente esquema. El orden de los mismos no tiene porqué ser el mismo que se muestra en el esquema, pero de aparecer, tienen que ser únicos.

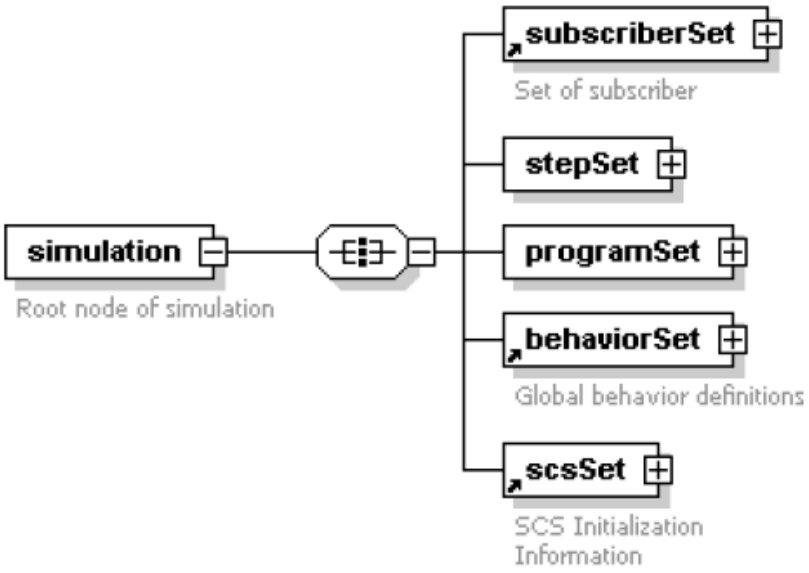

Figura 32. ***Etiqueta simulation***

Mediante **<suscriberSet>** definimos el conjunto de los abonados, **<suscriber>**, que van a estar presentes en la simulación. Se define para cada uno de ellos su nombre, mediante un atributo y su MSISDN (también es posible definir un alias). Otro posible atributo es *groupSize*, por defecto a uno, y que indica si estamos ante un abonado o un grupo. Se ilustra con el siguiente esquema:

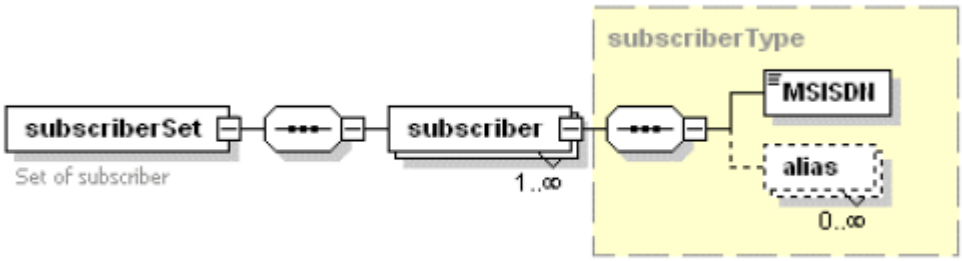

Figura 33. ***Etiqueta suscriberSet.***

Con la etiqueta **<stepSet>** definimos un conjunto de pasos. Cada uno de ellos se define mediante la etiqueta **<step>**. Cada script puede definir uno o más programas. Un programa no es más que referencias ordenadas a pasos definidos dentro de <stepSet>. Estos pasos son distintos según a que SCS pertenezca la actividad que queremos realizar. También pueden definirse pasos globales, que afectarán a todos los SCSs (como añadir o remover abonados).

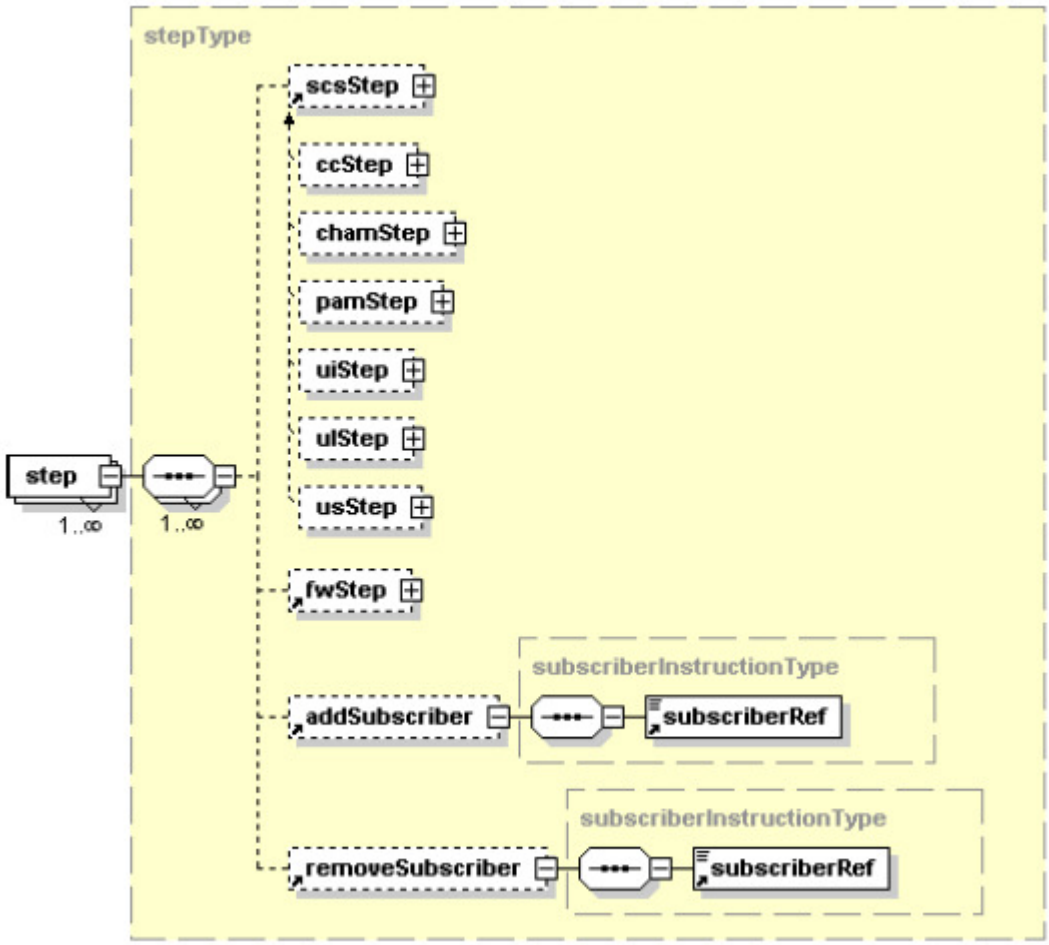

Figura 34. ***Etiqueta step***

Vemos como la etiqueta <scsStep> que aparece en el esquema anterior es abstracta, es decir, sólo podemos usar una concreta: **ccStep**, **ulStep**,…, dependiendo de a qué SCS pertenezca la acción.

A la etiqueta <step> se le puede especificar un nombre, mediante el atributo *name,* que será la forma de referenciar dicho paso en cualquier punto del script.

Como se ha comentado anteriormente, en la definición de los programas, lo único que se hace es referenciar pasos definidos anteriormente, y asignarles un orden aportando también información de tiempo. Con la etiqueta **<programSet>** se engloban todas las definiciones relacionadas con los programas. Cada programa que se defina ha

de estar encerrado entre etiquetas **<program>**, y con **<programStep>** se referencia el paso que se necesite. Se ilustra lo anterior con el esquema correspondiente.

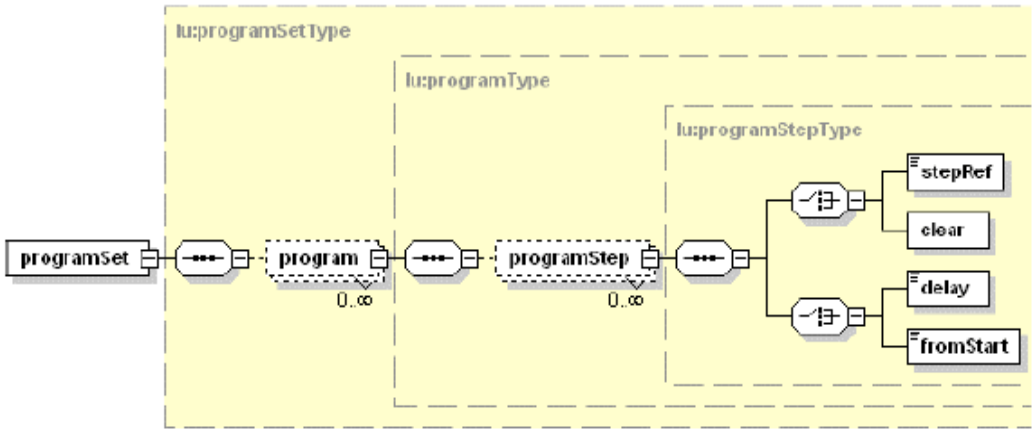

Figura 35. ***Etiqueta programSet***

Como se puede apreciar en la figura 36, para definir un paso de un programa, o bien se referencia un paso previamente definido, o bien usamos la etiqueta **<clear>**, que lo que hace es reiniciar todos los datos del simulador. Además, mediante la etiqueta <delay> podemos indicar el retraso en milisegundos con respecto al paso anterior, o bien con <fromStart> se indica el retraso en la ejecución del paso pero con respecto al inicio del programa. Para definir el orden se utiliza un atributo dentro de <programStep>, *seq*, que toma valores enteros.

Por último comentar que la etiqueta **<behaviorSet>** contiene una lista global de definiciones de comportamiento, diferente para cada uno de los SCSs. Cada SCS posee sus etiquetas propias. Se muestra el esquema general a continuación.

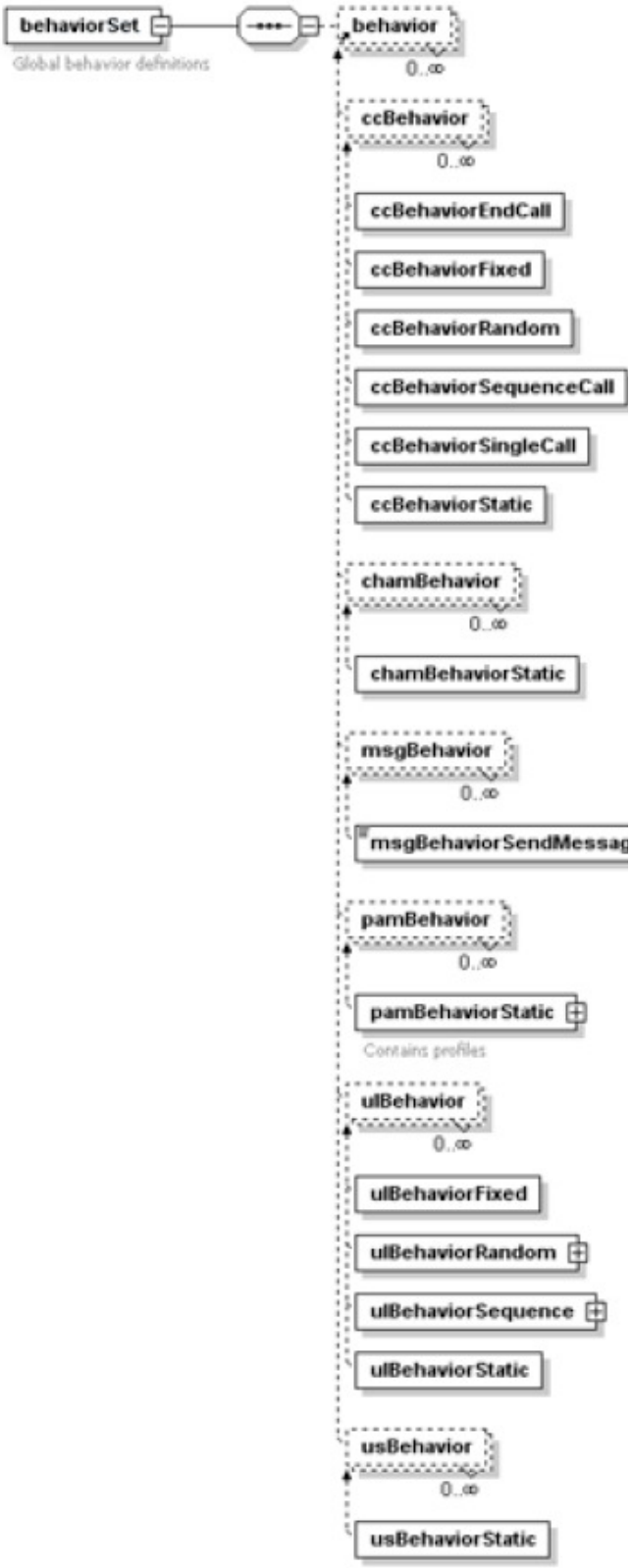

Figura 36. ***Etiqueta BehaviorSet***