Requirements for an Extendible IMS Client Framework

Andreas Bachmann
Fraunhofer Institute for Open Communication Systems (FOKUS)
Kaiserin-Augusta-Alle 31
10589 Berlin, Germany
+49 30 3463 7128
bachmann@fokus.fraunhofer.de

Alice Motanga
Fraunhofer Institute for Open Communication Systems (FOKUS)
Kaiserin-Augusta-Alle 31
10589 Berlin, Germany
+49 30 3463 7335
motanga @fokus.fraunhofer.de

Thomas Magedanz
Technische Universität Berlin / FOKUS
Kaiserin-Augusta-Alle 31
10589 Berlin, Germany
+49 30 3463 7229
magedanz@fokus.fraunhofer.de

ABSTRACT

As IMS matures, the industry and research institutes are steadily adopting IP Multimedia Subsystem (IMS) by building internal IMS playgrounds for prototyping and validation. From all the IMS components, the IMS User Equipment (UE) is a critical entity for the overall success in the IMS value chain. This is because, the UE is the only component that demonstrates IMS services found on the network, and the presentation of these services to the end-user will determine the return on investments on IMS. Early IMS service demonstrations are standalone solutions or support a limited set of basic services such as Voice over IP (VoIP), Presence and Messaging with little or no room for extensibility or reusability. The objective of this paper is to analyze and report our activities to develop an IMS client framework that provides intrinsic IMS functionalities and supports reusability, service composition/aggregation for seamless user experience, extensibility and dynamic service provisioning.

***Index Terms* — IP Multimedia Subsystem, IMS Client, Framework architecture**

1. INTRODUCTION

IMS deployment is an integral part of the Next Generation Networks in Europe and other parts of the world. In the past few years, there has been an uptake of IMS test-beds and playgrounds, demonstrating the capability set of IMS. However, more focus until now has been predominantly laid on platform architecture and service design from the back end and service integration perspective. Currently available number of IMS client and client framework indicate that the IMS standardization efforts in UE have led to lack of available IMS compliant and island solutions leading to interoperability problems. This, is due to three main reasons; incomplete or missing specification of IMS standards in client development, the limitation of IMS services to VoIP together with presence and messaging, limitation in solutions leveraging a large scale of end devices. Current IMS client solutions limit the understanding of IMS services to offering network controlled multimedia services by combining voice and data in a packet switched network, but ignore to see beyond that, for example the use of IMS to develop a service environment to leverage innovation of Web 2.0 and social networking of user generated content. The market potential focuses mostly on mobile operators, leaving out IMS services for fixed line consumers or cable operators. This results in a poor availability of IMS clients and interoperability problems on different IMS networks. A more efficient approach would be an open specification for an IMS client framework, on which to develop IMS and non IMS-based services with a uniform user experience to run on different networks and on a large scale of end devices (desktops, PCs, mobile devices, set-top boxes, home gateways etc.).

This paper analyses the current works in standardizing IMS client architecture and presents our activities in developing and validation of our Open IMS Client framework. The framework provides development of applications in a fixed-mobile convergent environment, interoperability with open standard IMS networks, extensible for future applications and easy update management to existing components. Figure 1 depicts the components of FOKUS IMS playground. Our IMS playground comprises of the following elements;

FOKUS IMS Core: comprises of the Call State Control Functions (Proxy/Interrogating/Service CSCF) and the Home Subscriber Server (HSS). The CSCFs provide a set of IMS compliant intrinsic components specified in 3rd Generation Partnership Project (3GPP) release 6 [1] for routing, service discovery, charging amongst others. The HSS is a database to store user profile and service data.

SIP Servlet Execution Environment (SIPSEE): is FOKUS development of a SIP application server based on SIP servlet technology. SIPSEE provides multimedia session control capabilities to converged (IMS and HTTP-based) multimedia services.

Open Communication Server for Parlay X (OCS-X): is a FOKUS implementation of a Parlay X web service specification for telecommunication networks. OCS-X provides a network abstraction API for 3rd party service provider application docking unto the IMS network.

XML Data Management Server: is a data server to store service centric, user centric and device centric configuration data. The data on the server can be managed from the client or other network entities, using XML Configuration Access Protocol (XCAP) [2].

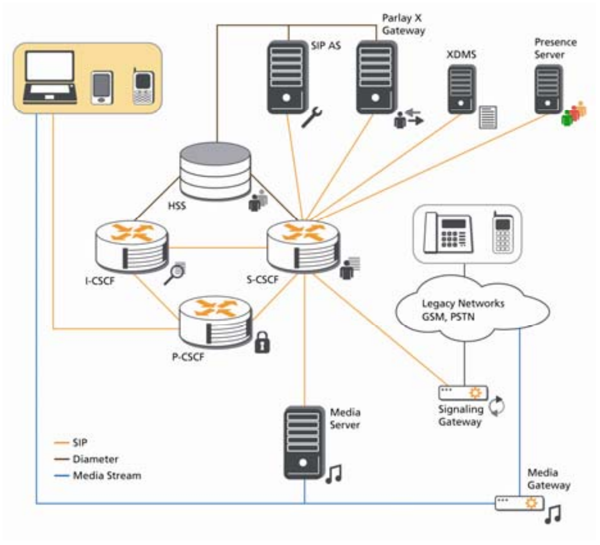

Figure 1. FOKUS IMS Playground

Presence Server: is a FOKUS implementation of an IETF-Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) compliant server for managing user's Presence Information (PI) document. Applications can use the PI not only to see when friends or contacts are online, but also to enhance multimedia services with context information that can be used to develop rich user experience

Media server: is an open standard industry partner solution that provides media announcements and media conferencing functionalities.

Signaling and Media Gateways: are open standard industrial gateway solutions for interoperability with legacy networks (e.g. GSM, PSTN).

Open IMS Client (OpenIC): is FOKUS implementation of an IMS UE based on our IMS Client framework to prototype the multimedia services existing on the FOKUS IMS Playground. The client runs on multiple device platforms (pocket PCs, PCs, laptops) and on multiple runtime environments (Windows Mobile, Windows XP/Vista and Linux).

This paper addresses the research activities of IMS Client architecture specifications. It describes the design and vision of an open IMS Client framework that provides intrinsic IMS application enabler functionalities for developers to build IMS and non-IMS based services. The framework hides the complexity of the IMS technology, allowing the developer to worry only about developing applications for the IMS network. It also offers service reusability as well as service composition to build rich applications. We envision the framework also to provide management tools, which makes it possible for remote update of components, dynamic deployments of new services and personalization of available services. This way, operators can deploy IMS client solutions and have the possibility of updating new releases of service components. The paper also describes the development of a prototype application as proof of concept to validate the framework.

2. RELATED WORKS

The development of our IMS Client framework solution is not intended to specify competing standards to already existing standards, but to use and refer to available standards where needed. There already has been work done in specifying service enabler architectures for IMS services, which include the following;

3GPP Release 5 [6], which specifies the IP Multimedia Subsystem (IMS) and Release 6 [1], which specifies the second phase of IMS, define some multimedia broadcast and multicast services applicable to UMTS and GSM networks, which was later adopted by 3GPP2 and TISPAN. Global Certification Forum (GCF) and PCS Type Certification Review Board (PTCRB) base their interoperability testing on these specifications. The aim of 3GPP specifications concentrate on the IMS as the signaling platform rather than a service delivery platform above IMS. However, the specifications help in defining the scope of services applicable, subsequently acting as a guideline in defining IMS services and developing IMS clients.

Open Mobile Alliance (OMA) [11] is an organization that specifies mobile data services by specifying market driven service enablers on top of IMS. Their specifications ensure service interoperability across open standard implementations clients, service providers, and operators as well as IMS networks. Extensive work has already been specified for various service enablers with the OMA service environment that specifies client and service behavior for service implementations, which can to some extent be applied in client development. The works from OMA lays the ground requirements for identifying functionalities for the different application enablers on our client framework.

The Java Specification Requests (JSR) 281 [7] define mobile device specific platform IMS client framework on which operators and third party service providers can develop and provide new IMS services. JSR 281 specification exposes IMS functionality through high-level APIs that hides IMS implementation details through abstraction of the underlying technology. This approach secures conformance to IMS related standards and at the same time gives developers possibility to focus on the functionality of the services and not on the IMS technology implementation details. JSR 281 specifies a high-level definition of the client architecture and service enabler. Nevertheless, no reference architecture as proof-of-concept has been deployed. In addition, JSR 281 framework specifications concentrated solely on mobile end devices for mobile operators, ignoring fixed and cable operators, who can also enhance IMS to provide innovative multimedia services to their subscribers.

However, as mentioned at the beginning of this section, our activities on developing an IMS Client framework is not intended to specify competing standards to already existing standards or approaches, rather to use and refer to them where applicable.

3. CLIENT APPLICATION PRINCIPLES

Same as any architecture, the client architecture for IMS consists of a set of layered and structured modules, with each module comprising of well-defined functionalities. Figure 2 illustrates a high-level architecture description of the framework. The framework comprises of the application layer and the service layer.

The application layer comprises all the IMS and non-IMS applications hosted on the client. The service enabler layer comprises of service enablers responsible for providing service intrinsic functionalities. Intrinsic functions are those functions, which are essential in fulfilling the intended task of the specified service enabler. A service enabler implementation is an implementation of a related set of functions that perform useful work, enabling one or more services. Other service enablers within the framework can as well use these enablers.

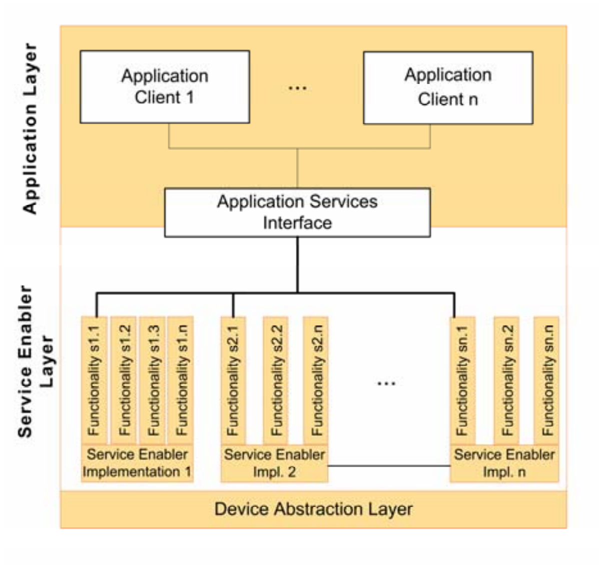

Figure 2. IMS Client Architecture

The following sub sections describe the key principles of the IMS Client framework.

3.1 Intrinsic functionality

As mentioned above, intrinsic functions are functions that are essential in fulfilling the intended task of the specified enabler. For example, Authentication is intrinsic to Registration, just like Registration is intrinsic to publishing Presence Information. Non-Intrinsic functions are those functions that are not essential in fulfilling the intended task of a specified enabler. For example, publishing Presence Information is a non-intrinsic function to establishing an audio session with a peer.

Any requirements or features that are not intrinsic to an enabler should not be specified within that enabler's specification. An enabler's specification should only specify the intrinsic functionalities required to fulfill its actual function. Therefore, the client framework should classify intrinsic and non-intrinsic functions for enablers relative to other service enablers.

3.2 Delegation and reuse of enablers

Service enabler specification should make use of already existing specifications where possible. For example, the reuse of presence and group list enablers for conference calls. In order to reuse functionalities from other service enablers, enabler specifications have to specify interfaces between service interfaces on how to use cross functionalities. This also enables a general horizontal, rather than a vertical architecture, that acts as a docking station for applications. A service enabler implementation can invoke existing standard functions, such as authentication or group management that it needs to satisfy its intrinsic functions defined in its specification.

3.3 Extensibility

Within an IMS Client framework, specifications of service enablers expose standard interfaces for IMS applications and that other service enablers can use as well. These service enabler interfaces, connect to actual service enabler implementations within the framework. Through this abstraction, it is possible to add or modify the underlying implementations without affecting the interfaces exposed by the enabler specification. This way, the framework allows introduction of new functionalities.

3.4 Management

Management refers to the process of managing framework settings and applications resident on the framework. This feature is applicable in various scenarios. For example, remote service provisioning gives operators the possibility of querying device capabilities, updating, deploying and maintaining IMS client applications and framework components. The user can also use the management tools to personalize the framework with a set of available applications, or manage the different application preference settings for the available services.

4. OPEN IMS CLIENT FRAMEWORK

To reach the goal of creating an open, extensible framework, we separated the functionality into different components. Each of these components implements a specific part of the framework stack. The framework enables application developers to develop IMS and non-based IMS applications, making use of underlying IMS intrinsic functionalities for defined services. This approach secures conformance to IMS related standards and at the same time gives developers possibility to focus on the functionality of the services and not on the IMS technology implementation details. In addition, developers can use existing modules, to build rich applications, by aggregating services from multiple modules. For example, an IPTV application can integrate watching TV or Video-on-Demand functionality together with the call, presence and contact list service modules to receive notifications, when a friend logs online or for incoming calls. The module-based approach also enables features like;

- automatic update for parts of the framework or the resulting applications
- implementing different functionality for specific devices
- co-location of multiple IMS Services
- encapsulation of internal protocols

To keep the framework maintainable we made the modules self-describing. Each module contains a name, a small description, a version and declares its dependencies to other modules. This will allow the framework to load dependent modules automatically and avoids circular references during development. For rapid application prototyping, it is important to reduce the amount of code, which the developers must write for application development. We decided to create several layers of abstraction by grouping different modules with similar functionalities and grouping intrinsic functionalities used by other modules, such as registration.

The framework is divided into four different layers namely the device abstraction layer, the network layer, the application enabler layer, and the application layer. Developers will have the choice on which layer they start to build the application. Developing on a lower layer increases the effort for building new applications but means more control. The following sections define the different framework layers.

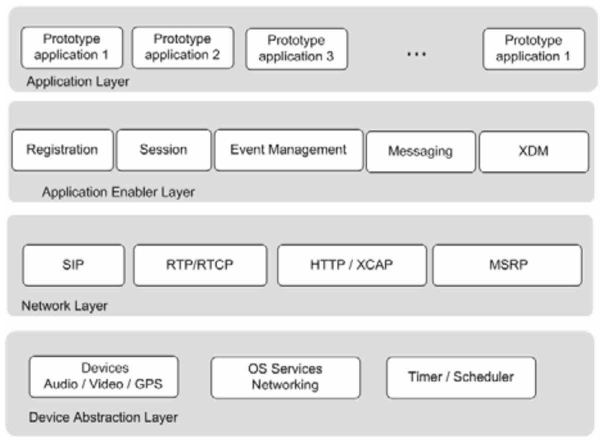

Figure 3. Client Service Creation Environment

4.1 Device abstraction layer

The heading of subsections should be in Times New Roman 12-point bold with only the initial letters capitalized. (Note: For subsections and subsubsections, a word like *the* or *a* is not capitalized unless it is the first word of the header.)

4.2 Network layer

The network protocol layer provides access to the different protocol stack APIs used for IMS and multimedia services. There are components for SIP, RTP/RTCP, HTTP, XCAP and MSRP. The framework uses the SIP stack to develop IMS compliant SIP applications that are compliant with IETF, 3GPP and TISPAN IMS standards. It provides a low level API for full control over SIP communication between the client and IMS network. The RTP/RTCP module provides the protocol stack for real-time audio/video communication, encoding and decoding functionalities for different codec and media control. The framework uses the HTTP stack to establish HTTP connection sockets to the Internet. The XCAP module also uses it, in order to establish connection to XDM servers. The Message Session Relay Protocol (MSRP) implements the stack for transmitting near real-time, peer-to-peer exchanges of binary content. MSRP uses a rendezvous protocol such as SIP for signaling

4.3 Application enabler layer

The application enabler layer combines the protocol components and the device layer components to form the core of the framework that enables plug-in or development of different applications. This layer hides the complex signaling of IMS. It is the toolkit used by all applications (IMS and non-IMS) to provide defined services. The application enablers comprise the following modules:

4.3.1 Registration

Before accessing any services on the IMS network, the UE first needs to register and bind its address on the network, which the network uses in routing messages to the UE. This enabler provides registration and authentication functionalities to the above applications. The enabler implementation includes two HTTP Digest authentication algorithms; Authentication and Key Agreement and Digest Access authentication scheme using MD5. In addition to the authentication algorithms, this enabler implementation also includes SIP “path” extension header field for registering adjacent contacts and uses the SIP extension header in the registration response from the server for service route discovery.

4.3.2 Event management

This enabler implements an extensible event framework where applications can request notifications from remote peers or IMS entities, indicating that certain events have occurred. This event framework enabler provides SUBSCRIBE and NOTIFY methods to different event packages. The supported event packages include “refer”, “reg”, “presence”, “winfo” , “ua-profile”, “conference”.

4.3.3 Messaging

This enabler provides pager-mode based, as well as session-mode based advanced messaging sessions. Application can use this enabler to send instant messages, start a chat session with a remote peer, or send binary files to a remote peer.

4.3.4 Session

The session enabler provides applications with an abstract point-to-point multimedia communication with a remote peer. The communication comprises a set of media connections types (audio, video and message). It uses the Session Description Protocol (SDP) to describe and negotiate the connection details for the different media types. After the session is established, it offers also the possibility of modifying the session. For example, to add or remove media types to /from the communication. This enabler also provides the possibility of transferring the session to a different device (also known as Explicit Communication Transfer).

4.3.5 XML data management (XDM):

The XDM enabler provides applications with a document framework for managing and manipulating XML documents as specified by OMA [4]. The XDM module offers XML object serialization functionalities to parse documents to objects. The enabler also provide applications the possibility to create XCAP connections to the XDMS on the network, to store, configure and manage service data on the network.

4.4 Application layer

The application layer is where developers can develop and include basic and innovation applications that use the application enabler framework. The applications contain views for user interaction. Views are presentation specific implementations of user interaction objects. They are located in own modules and are specific to the device the client is running on.

The next sections describes some of the prototype applications we have developed using the Open IMS framework as proof-of-concept solutions.

5. PROTOTYPE APPLICATION

As proof-of-concept for our work in developing the Open IMS Client Framework, we implemented prototype applications ranging from simple basic IMS application, to innovative service composition applications, which demonstrate reusability of service components and their composition to form new applications. Figure 6 illustrates the different applications currently resident on the framework. Our solutions run on Pocket PC as well as desktop/laptop PCs. In this section, we shall concentrate on the details of one application from the set of capabilities. The rest of this section describes the location-based service.

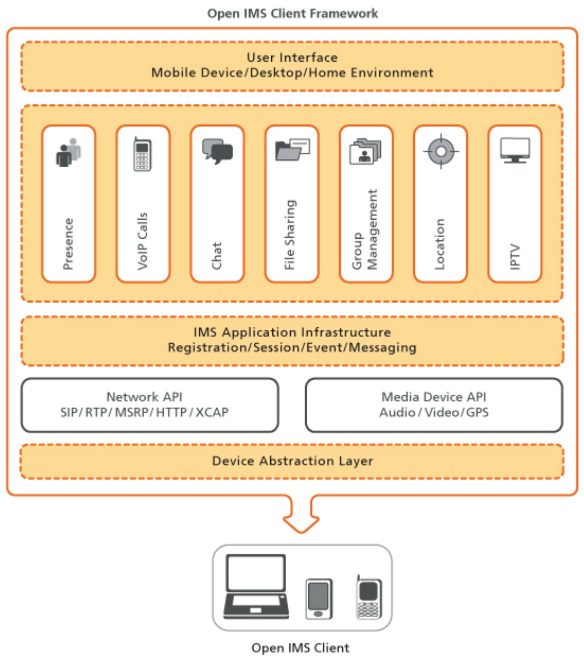

Figure 4. Application prototypes on OpenIC framework

Location services are services based on Location Information (LI). Subscriber services can be enhanced with the use of LI. For example, a subscriber's mobile set can be provisioned on the fly with new services, based on his/he current location (find the nearest restaurant, petrol station etc.). LI can as well be used for emergency applications to emergency call to the nearest police station, hospital etc. These are a few examples how location can be used to enhance applications. We use the same concept of GI to create a rich application for sharing contacts geographical information using Presence.

Next, we shall explain the technical prerequisites for such an application. Since the application is based on GI, it requires a source for acquiring location information. There are several techniques available as sources for acquiring GI such as Location beacons (using Bluetooth), subscribing to carrier information, or manually configuring the location. However, we made use of the Global Positioning System (GPS), because it is widely used and widely available on today's mobile devices. Another prerequisite we considered for the developing the contacts geo presence application is privacy. The user should have the possibility to choose if he/she wants to communicate their locations for others to see. Last but not the least, in order to display users' information to the subscribed user, we included a map application on the framework, which graphically displays the exact location of any contacts user who chooses to publish their GI for others to see.

The development of the location based contacts presence application based on the geographical location and privacy specification and architecture [5] [8]. The four primary entities in Geopriv include the Location Generator, the Location Server, the Location Recipient and the Rule Maker.

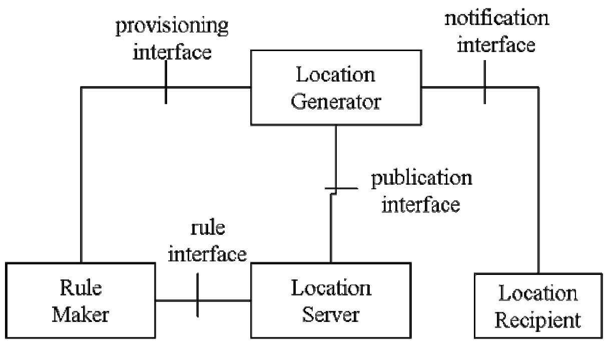

Figure 5. Geopriv entities

The Location Generator determines the geographical Location Object (LO) of an object that describes a location and publishes the LO to the Location server. Location server receives the LOs and receives subscriptions from various location recipients. In coordination with policies set be the Rule Maker, it distributes the location information to Location Recipients. The Location Recipient subscribes to LO on the Location Server ant renders notified event for LOs to a user or some receiving application The rule holder is the repository for privacy rules filtering and distributing LOs for specific entities. A user populates the rule holder with privacy rules

On our prototype scenario, the Presence Server and the XDMS on the FOKUS IMS Playground played the role of the Location Server and Rule Maker respectfully. For the Location Generator and the Location Recipient, we used two Open IMS Clients as UEs on Windows Mobile devices. The UEs use registration enablers to access the IMS network and the event framework to publish, subscribe to and receive notifications form the Presence Server. For specifying rules for privacy on the Rule Maker, the UEs use the XCAP handler to define rules for the Geopriv service, which the Presence server uses to dispatch information to subscribed recipients

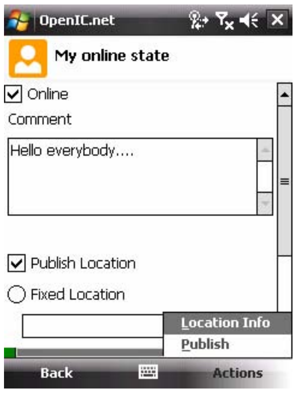

Figure 6. Configure and publish own configuration information

As Figure 8 depicts, the end user can configure their Presence Information with basic "online" "offline" status, a personalised message and geographical Location Information. After the user registers on the Open IMS network, a SIP PUBLISH is sent to the Presence Server with an extended PIDF document. The PIDF document contains Geography Markup Language (GML), which describes the geospatial information of the user. The geographical coordinates are read from the device's GPS receiver. Addition to the SIP PUBLISH message sent after registration, the contacts application sends a subscription message (SIP SUBSCRIBE) to the Presence Server for receiving presence information for contacts on the contact list. We extended the framework with a map application, which graphically depicts the GI of a contact on a map. The end user can view their location information, as well as that of their friends.

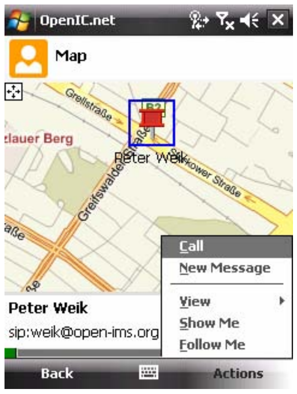

Figure 8. Map Application showing Contact's Geographical Location Information

From the map application, the end user communicates directly with the remote contact by placing a VoIP call or sending a message to the remote contact.

6. CONCLUSION

IMS adoption is slowly but steadily rolling out in internal laboratory test-beds for service prototype developments. Most of the attention until now was centered on the IMS network and the Service Delivery Platform for providing commercial IMS services. Little attention has been given to the IMS clients that play an important role in user acceptance for IMS services. There are no open standard specifications on developing services on IMS clients, which has lead to poor availability of IMS clients or interoperability problems. This paper presented an open extensible approach for an IMS Client framework, which refers to works from JSR 281 and client requirement on OMA service specifications. The framework allows rapid development and prototyping of IMS based client applications. The use of modules was a key concept during the design and the implementation phase. It allows the framework to be extended by third parties as to be adapted at specific runtime environment. The framework was implemented using Java and C#. The same concept can be transferred to any programming language. The current version of the framework is build in form of libraries which are linked together to form a specific application. Future versions of our framework is planned to be stand-alone IMS stack implemented on the operation system layer.

7. REFERENCES

[1] 3GPP Version TSG #33, "Overview of 3GPP Release 6", Release 6, 2006

[2] IETF, "RFC 4825 – The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", May 2007.

[3] FOKUS Open IP Multimedia Subsystem Playground http://www.fokus.fraunhofer.de/ims/

[4] OMA XDM Core, "XML Document Management (XDM) Specification", Approved Version 1.0 – 12 Jun 2006.

[5] IETF "RFC 4119 – A Presence-based GEOPRIV Location Object Format".

[6] 3GPP, "Overview of 3GPP Release 5, Summary of Release 5 Features", Release 5, September 2003.

[7] Sun Microsystems, "JSR 281 – IP Multimedia Subsystem (IMS) Services API", Public Draft version 0.9, September 2007

[8] H. Tschofenig, H. Schulzrinne, A. Newton, J. Peterson, A Mankin, The IETF Geopriv and presence architecture focusing on location privacy, Position paper at W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement, Ispra, Italy, 2006.

[9] O. Friedrich; A. Al-Hezmi; S. Arbanowski; T. Magedanz; "Next Generation IPTV services for an extended IMS architecture"; Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium on March 2007 Page(s):429 – 436, Digital Object Identifier 10.1109/ISADS.2007.52