Definition of a Web 2.0 Gateway for 3rd Party Service Access to Next Generation Networks

N. Blum[*], D. Linner[*†], S. Krüssel[*], T. Magedanz[*†], S. Steglich[*†]

[*]Fraunhofer Institute for Open Communication Systems (FOKUS),
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{niklas.blum|david.linner|steffen.kruessel|thomas.magedanz}@fokus.fraunhofer.de

[†]Technische Universität Berlin,
Sekr. FR 5-14, Franklinstrasse 28/29, 10587 Berlin, Germany
{thomas.magedanz|stephan.steglich}@tu-berlin.de

Abstract Modern telecommunication networks and classical roles of operators are subject to fundamental change. Many network operators are currently seeking for new sources to generate revenue by exposing network capabilities to 3rd party service providers. At the same time we can observe that applications on the World Wide Web (WWW) are becoming more mature in terms of the definition of APIs that are offered towards other services. The combinations of those services are commonly referred to as Web 2.0 mash-ups. This report describes our approach to include Next Generation Networks (NGN)-based telecommunications application enabler into Web 2.0 mash-ups by defining a JavaScript-based service exposure API that allows easy and straight forward integration of telecommunications enablers into such mash-ups. The platform is validated through an application including telecommunications-based services as conferencing, rich presence and location, as well as the community portal Facebook and Google maps.

1 Introduction

Telecommunications is at crossroads, the convergence of fixed and mobile telecommunications, cable networks, as well as the Internet leads into a global all-IP based Next Generation Network (NGN). Through this ongoing process of the convergence of access networks and the existence of new players in the telecommunications market, traditional operators and carriers are seeking for new business models to increase their revenue. The reuse of an extensible set of existing service components to rapidly create new market driven applications is a key aspect of telecommunications platforms since many years and gains a new momentum with the definition of dedicated application enablers for NGNs. One real-life example is

British Telecom's BT Web21C SDK [1] solution that defines an API to expose telecommunications specific core network functionalities to 3rd party service developers using Web Services.

This paper describes our approach of the realization of a service access gateway for applications based on a JavaScript API to address the specific needs of Web 2.0 developers. It is validated within a prototyped application for the community portal Facebook [2] offering access to functionalities such as conference management, location, SMS, MMS and rich presence.

The paper is structured as follows: Section 2 provides a brief state of the art overview of the NGN functionality namely the IP Multimedia Subsystem (IMS) with focus on application enablers and technologies associated to the term Web 2.0. Section 3 describes our concept of a service enabler access gateway for Web 2.0 mash-ups. Section 4 provides an overview of the prototype application. We end the paper with a conclusion and outlook in section 5.

2 Related Standards and technology Overview

The following subsections describe emerging standards as the IMS, IMS enablers, related technologies to the term Web 2.0 like Ajax and the mash-up service architectures. Furthermore it shortly depicts existing solutions in the area of service exposure for Web-based applications.

2.1 The IP Multimedia Subsystem

The IP Multimedia Subsystem (IMS) [3] has been defined from the 3rd Generation Partnership Project (3GPP) Release 5 specifications on as an overlay architecture on top of the 3GPP Packet Switched (PS) Core Network for the provisioning of real time multi-media services. It is based on Internet Engineering Task Force (IETF) protocols like the Session Initiation Protocol (SIP) [4] for session control and Diameter [5] for Authentication, Authorization and Accounting (AAA) and charging purposes. The basic IMS architecture is depicted in figure 1.

Due to the fact that the IMS overlay architecture is widely abstracted from the air interfaces, the IMS can be used for any mobile access network technology as well as for fixed line and cable access technology as currently promoted by ETSI TISPAN (TIPHON (Telecommunications and Internet Protocol Harmonization over Networks) and SPAN (Services and Protocols for Advanced Networks)) within the Next Generation Network (NGN) reference architecture definition [6].

The IMS provides easy and efficient ways to integrate different services, even from third parties. Interactions between different value-added services are anticipated. It enables the seamless integration of legacy services and is designed for consistent interactions with circuit switched domains. Furthermore it supports a mechanism to negotiate Quality of Service (QoS) in different access networks. The IMS also provides appropriate charging mechanisms for online and offline charging. Thus you can realize different business models and charge for specific events using an appropriate scheme.

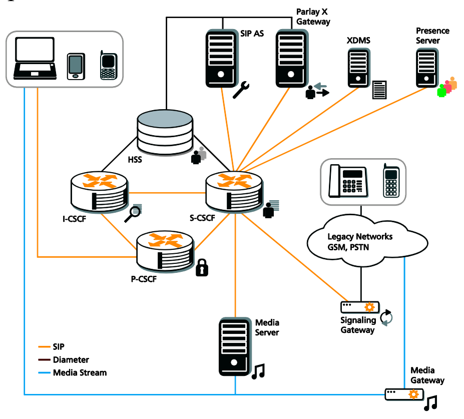

Fig. 1 Basic IMS Architecture

The particular techniques and methodologies that are required to gain the advantages of these key functionalities are not completely new, but the IMS provides the first major integration and the interaction of all key functionalities.

2.2 IMS Enabler

Similar to Service-Independent Building Blocks (SIBs) which form part of the conceptual model for Intelligent Networks, the Open Mobile Alliance (OMA) defined during the last years service enablers for the IP Multimedia Subsystem. The ideas was initially born during the specification of a Push-to-Talk over Cellular (PoC) [7] service, a walkie-talkie like communication service between several mobile peers based on the Internet Protocol (IP) using the SIP, Real-time Transfer Protocol (RTP) and Real-time Transfer Control Protocol (RTCP). PoC uses Presence, Group Management and Instant Messaging as enablers to provide information to the users as well as to the PoC service. This led alongside the standardization of PoC to the definition of Presence SIMPLE [8] for Presence and Instant Messaging and XML Documents Management (XDM) [9] for group and list management.

PoC as a public available service never received real acceptance besides the U.S. market, but the concept of abstract application enablers is by now widely used.

Service developers for next generation network based applications, especially those offered by 3rd party service providers will want to make use of the advanced multimedia communication functionalities offered by IMS-based applications. But core communication functionality like voice- and video call control as well as legacy messaging and location will reside at the operator's domain for security reasons and a well-defined integration of the service platforms into the operator's charging and provisioning functionality. Most application developers will also not have the capability and resources to economically develop such complex communication features into their services. The OMA currently standardizes the OMA Service Environment (OSE) [10] as an abstract enabler layer that serves as an access gateway for 3rd parties and operator services. Figure 2 depicts the OSE architecture:

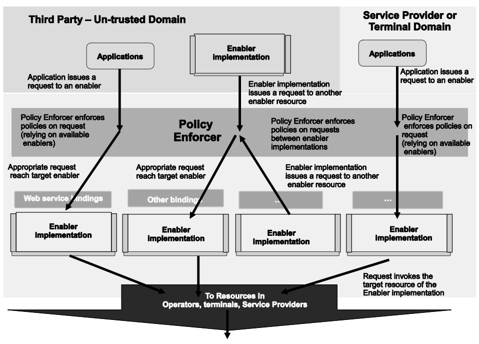
Third Party – Un-trusted Domain
Service Provider or Terminal Domain
Applications
Enabler implementation
Applications
Application issues a request to an enabler
Enabler implementation issues a request to another enabler resource
Application issues a request to an enabler
Policy Enforcer enforces policies on request (relying on available enablers)
Policy Enforcer
Policy Enforcer enforces policies on requests between enabler implementations
Policy Enforcer enforces policies on request (relying on available enablers)
Appropriate request reach target enabler
Appropriate request reach target enabler
Enabler implementation issues a request to another enabler resource
Web service bindings
Other bindings
...
...
Enabler implementation
Enabler implementation
Enabler implementation
Enabler implementation
Request invokes the target resource of the Enabler implementation
To Resources in Operators, terminals, Service Providers

Fig. 2 OSE Architecture.

The OSE has been introduced to enable operators with the functionality to provide their communication and application capabilities to users without the need for the application developer to implement such functionality into their applications themselves. The OSE provides various functions such as process monitoring, software life cycle management, system support, operation, management and administration controls the enablers

2.3 The WWW and Web 2.0

The WWW is by nature community-driven, not only with regard to content, but also from a technical point of view. Simple protocols such as HTTP, description languages as HTML and CSS, and architecture paradigms (e.g. Representational State Transfer - REST) made the Web successful and simplicity is the decisive factor for the developer community's acceptance of extensions to the Web technology stack. Web 2.0 is less a question of novel technologies in the Web technology stack, but rather a question of how existing technologies are applied to create services tailored to user communities.

In this respect, client-side active scripting and the inherent capability of HTML to integrate content from different sources play a major role. Active scripts are shipped along with the web content to control content presentation and interactivity. The object based programming language ECMAScript [11], better known as JavaScript, is today's mostly used scripting language for Web pages. In addition to operations on the associated document, all noteworthy Web browsers allow active scripts to self-reliantly utilize the HTTP client interface in a pared-down configuration. This feature of active scripts to access their origin server for the exchange any of messages is referred to as Asynchronous Java Script (Ajax) [12]. Although the Ajax API introduces with the XMLHttpRequest [13] just one new language construct the amount of available developer tools based on Ajax show its current importance.

The varieties of client-server interaction, given to active scripts through Ajax include Remote Procedure Call (RPC) and Publish-Subscribe. The representatives for RPC over HTTP in favor of the developer community are XML-RPC [14] and JSON-RPC [15]. The major difference between both can be found in the representation of request and response, i.e. marshalling of method calls and objects. While JSON-RPC utilizes a light-weight, non-standard syntax, XML-RPC is based on W3C's XML. However, RPC frameworks for Ajax usually require a respective counterpart on the server-side. In practice, tool support for the selected backend platform (e.g. .NET, J2EE, PHP) is often the decisive criterion for the selection of a RPC framework.

The same holds for message passing and publish-subscribe approaches based on Ajax and the support of HTTP 1.1 for continuous connections. Respective development patterns and the message protocol Bayeux [16] are summarized to a concept named Comet. Bayeux messages are represented in JSON (JavaScript Object Notation) syntax and marshaled within the entity bodies of HTTP requests and responses. Comet allows a Web server to notify events almost synchronous to Web page embedded active scripts. Hence, applying Comet within a Web application has a positive impact on the application's interactivity, which is especially appreciable in a Web 2.0 context.

The above technologies comprise a powerful client-side foundation for a novel approach of creating web applications, called mash-up. A mash-up is a composition of 3rd party service building blocks to a new, customized web application. Examples for such 3rd party service building blocks can be found in the open APIs of Google Maps [17], Yahoo search [18], Youtube [19], or Facebook [20]. While the outgoing web server of a mash-up provides the description of the composition and thus the actual services adding value to the building blocks, the rendering and execution of the mash-up happens at the client-side. Consequently, mash-ups potentially decrease the need for intelligent web servers and avoid bottle necks, since utilized 3rd party service building blocks are accessed by the mash-up clients directly. Furthermore, mash-ups enable the rapid creation of powerful applications, even with limited engineering skills. By today, *programmableweb* [19] lists more than 2500 serious mash-up applications on the web, tendency growing. Our *Telco / Web 2.0 Gateway for 3rd Party Service Access to Next Generation Networks* aims at creating a novel application building block that brings telecommunication services to mash-ups.

2.4 Existing Solutions / Related Work

With respect to the above mentioned correlation of simplicity and developer acceptance, a key objective of our efforts is to create opportunities for fast and easy customization of telecommunications services in the Web. British Telecom's Web21c [1] project follows a similar motivation, while providing APIs and respective SDKs to utilize services of their network within custom applications. The list of supported service comprises, messaging, voice call, conference call, authentication, inbound SMS, and call flow. The provided APIs are currently tailored to the integration with .NET and Java.

An interface to NGN services that is suitable for the creation of mash-ups is the SIPGate API [22]. Therein, XML-RPC is utilized to basically control calls and conference calls, obtain data from the phone book, check the account balance or the status of unified messaging.

A much smaller, but nonetheless charming exposition of a telephony network service for usage on the Web is realized by Skype4Web [23]. Web references to dynamic images allow obtaining users' presence states in the network. Skype also provides professional APIs to access network services such as call control, messaging, and presence, but due to the peer-to-peer nature of the underlying network the service access is integrated with the network client.

3 Telco / Web 2.0 Service Exposure Gateway

In this section we describe our design and implementation of a service enabler access gateway for Web 2.0 mash-ups using telecommunications enablers.

3.1 Parlay X Gateway

The OCS-X [24] is an implementation of the Parlay X Web Services specification for telecommunication networks. These interfaces provide a network abstraction through a very simple and easy to use API based on Web Services technology, which can be used remotely from 3rd party domains and service providers. The OCS-X uses the current Parlay X Version 2.2 [25]. Parlay X defines a set of powerful yet simple, highly abstracted, building blocks of telecom capabilities that developers and the IT community can both quickly comprehend and use to generate new applications. Each building block will be abstracted from the set of telecom capabilities exposed by the Parlay X APIs. The capabilities offered by a building block may be homogeneous (e.g. call control only) or heterogeneous (e.g. mobility and presence). A building block will usually not be application-specific. In order to use them from within the Web 2.0 domain a gateway has been developed, which allows access to the Parlay X API via JavaScript.

3.2 JSON-to-Web Services Gateway

Calling a Web Service from within an Ajax application is restricted to local Web Services, due to the Same Origin Policy enforced by modern web browsers. However, Web Services are used to be consumed beyond server limits at external endpoints. In order to call external Web Services from JavaScript, the service request has to be routed via some server component, which is called *gateway* in the following and is illustrated in figure 3. This gateway provides an interface for the client-side JavaScript, mapping it to the particular interface of the external Web Service. Thus, all parameters of an incoming Ajax request are passed on to the Web Service endpoint. The Java Script client does access a single server as usual, whereas the server may call Web Services remotely.

Web Services are typically accessed via SOAP messages that are difficult to handle with Java Script. The utilization of a gateway allows replacing SOAP by any protocol, as the protocol can be translated within the gateway. Therefore, more convenient data description formats, such as JSON and simple XML [26], can be used to ease the Web Service access.

Furthermore, the Web Service access can be simplified by abstracting the actual Web Service interface. Thereby, underlying Web Service business logic can be hidden from the Java Script developer as well as offered functionality can be expanded at the server side. For example, security constraints could be achieved by the gateway transparently.

In order to realize a gateway functionality in the mentioned fashion, different components are required that are depicted figure 3.

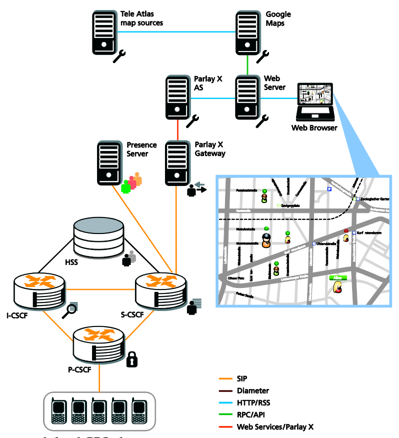

Fig. 3 Architecture of a telco-enabled Web.

The actual web application is stored on a regular web and connected to a database for consistent data management that is available via Ajax. Furthermore, 3rd party JavaScript APIs can be used within the web application (e.g. Google Maps) for mash-up.

In addition, an application server is required for the actual Web Service access. This server has to be connected to a Web Services platform offering multiple Web Services (e.g. OCS-X). The service access is again encapsulated within a simple API that can be used to extend existing Web 2.0 applications, such as the community portal Facebook. In order to use certain notification possibilities that could be offered by certain Web Services (e.g. in the IMS), Web Services endpoints have to be available on the gateway. Therefore, a Web Services engine as Apache Axis2 [27] has been established at the server-side. Moreover, the application server has to be connected to the Web Services engine to exchange incoming information passing it on to the web application via Ajax.

This configuration allows the fast development of complex mash-ups composed of Ajax APIs as well as open Web Services. Within the following section a concrete implementation of such a mash-up is introduced as a proof-of-concept.

3.3 Implementation of JSON-to-Parlay X Gateway

The realization of the above architecture has been transferred to Parlay X Web Services enabling IMS functionality. An overview about this concrete gateway is depicted in figure 4. The implementation is Java- and JavaScript based and the communication between client and gateway has been realized using JSON-RPC. Therefore, the open Java to JavaScript Object Broker (*jabsorb*) [28] is used on the client-side in order to transparently send JSON requests to remote Java objects. On the server-side, the jabsorb framework is used providing a particular Servlet that makes simple Java objects accessible via JSON-RPC that are automatically called on the accordant request.

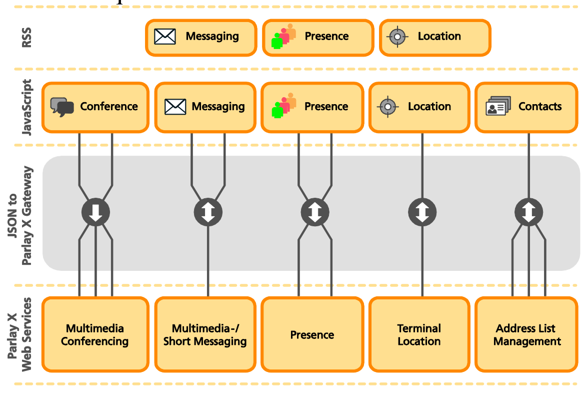

Fig. 4 JSON-to-ParlayX Gateway.

The Web Service access is done in each of those Java objects in order to request the required information. Indeed, the addressed IMS Enabler can be requested periodically to keep information up to date, it is intended that arising changes (e.g. location changes) are published automatically and not requested (pulled) manually. Therefore, the gateway must provide a Web Services endpoint itself to be notified from the Parlay X gateway. These Web Services are setup on the Axis2 engine [27] with their interfaces followed the Parlay X specification [25].

However, the classical Web model only allows for periodic polling, which makes server-push communications difficult. In order to make directly use of the notification mechanisms provided by the IMS the JavaScript client has to be extended by the Comet approach, which has been realized with the help of the HTTP-based publish/subscribe framework Pushlets [29]. This framework enables a direct forward of notifications to the client without periodic polling.

In order to forward incoming notifications, they must be available in the web server, since it holds the only connection to the client. Thus, the web server is connected to the Web Services engine that retrieves the particular information. The connection between web server and Web Services engine is done via cross-context communication [30] to share information between different web containers.

Based on this gateway, typical IMS features, like Instant Messaging, Conference Management and Rich Presence have been integrated into one of the largest online community platforms on the Internet – Facebook. A typical Web 2.0 mash-up composed of Facebook-Userdata, Google Maps and IMS Core functionalities emerged that demonstrates the easy and fast usage of the created gateway and its use for modern web development.

Facebook itself renders the development of a community application without building up the entire community platform from scratch. Moreover, already existing personal data had been easily integrated for further accretion. The visual presentation of location and presence data has been realized with the help of the Google Maps API. While Facebook is used through a PHP API, Google Maps is a pure JavaScript library that are both described in more detail in the following section.

4 Demonstrator / Validation

The demonstration of our Telco / Web 2.0 service exposure gateway, introduced above, is based on one of the largest Internet community portals with almost 65 million active users so far – Facebook. Facebook provides an API that allows developer to easily plug their applications into the portal getting access to Facebook user data, for example, retrieving information about all buddies of a user. The access on community-related information allows an easy mash-up of telecommunications features with user data. However, other APIs, such as Googles OpenSocial [31] could have also been used, but has been missed out due to its immatureness at the time of development. An overview is depicted in figure 5 that shows the current application and its embedding into the IMS infrastructure via the developed gateway.

In order to demonstrate a meaningful way to use information from within the IMS a map has been integrated into the application to present presence and location information in the Web. For this integration the Google Maps API has been used, but could have been exchanged by any other map API (e.g. Yahoo Maps). Additionally, user information from Facebook has been merged into the map to follow the Web2.0 idea of information mash-up.

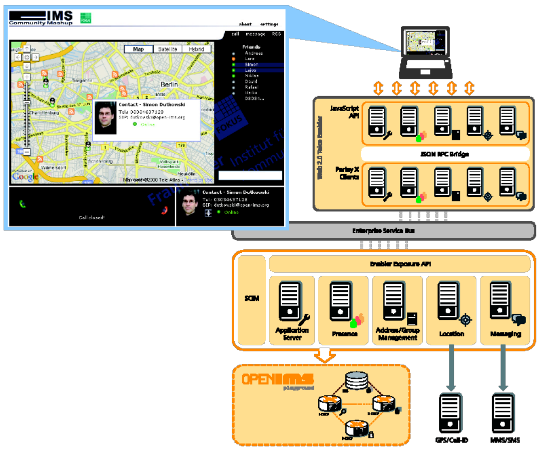

Fig. 5 Architecture of the Facebook Demonstrator.

Telecommunications specific-features have been realized in the following modules that each encapsulates a single functionality:

- **Multimedia Conference**: Voice conferencing through IMS. Participants can be contacted within the Public Switched Telephone Network (PSTN) as well as all-IP Next Generation Networks (NGN).
- **Presence**: Rich presence information, such as online/offline status and mood, of a certain user is published.
- **Terminal Location**: Request location information of a certain user from the IMS. It is necessary that the user provides location information from the IMS client, for example, with the help of a GPS receiver.
- **Short Messaging**: Sending simple text messages (SMS).
- **Multimedia Messaging**: Sending and receiving multimedia messages (MMS), consisting of plain text as well as application data. These messages are sent instantly and are also used for Instant Messaging (IM). A feedback channel has been also realized for this module to receive incoming messages.
- **Address List Management**: Transfer existing user- and group-information from the Web- to the Telco-domain and vice versa.

As a typical Web 2.0 feature, the integration of News Feeds has been also realized. However, the concept has been expanded on the telco side. The user can subscribe to different feeds that are collected within a single feed. The user than is being informed about every new entry. Furthermore, this feed is extended by location information out of the IMS. Therefore, user data at Facebook can be combined by interesting places.

Nevertheless, the developed application does not replace an IMS client, since it is only possible to handle certain management issues (e.g. call setup) or simple functions (e.g. messaging), but it is not possible to act as a communication end-point sending and receiving RTP streams. In fact, a typical IMS client is necessary for the provisioning of location and presence information as well as for the actual call setup. Therefore, the OpenIC IMS client [32] that is illustrated in figure 6 has been extended by Web 2.0 features like a RSS module for writing own RSS feeds as well as a map module that shows one's current location in a Web 2.0 like fashion. Moreover, all information published by the Address List Management component of the gateway can be received by the client as well, which contains information about the friends of a Facebook user and makes that information available in the IMS. Finally, an auto-provisioning mechanism has been implemented that allows for the subsequent integration of client modules triggered from within the Web.

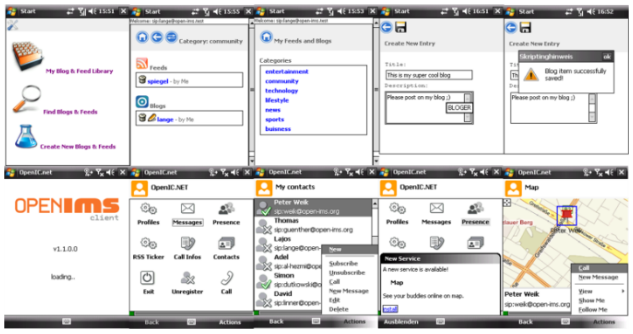

Fig. 6 Web 2.0 enabled IMS Client.

5 Conclusions and outlook

In this report, we described our implementation of a Web 2.0 API gateway for telecommunications services and a proof-of-concept integration as a Facebook application. Our focus was on the creation of a very high level API that meets the needs and programming paradigms of developers from the WWW community. On the other hand it was important to us to integrate and achieve full interworking with existing legacy devices. Another emphasis was the reuse of existing IMS enablers and to keep the implementation as generic as possible in regard of reusability of the components.

Architecture-wise, our service infrastructure is acting on top of a mobile operator infrastructure and offers operator core services like SMS and call control with enhancements through Web 2.0 features on the presentation layer provided by a mash-up. The implementation is part of the Open SOA Telco Playground at

Fraunhofer FOKUS [33] and provides the Web 2.0 Telco Enablers and the Web 2.0 API gateway.

Future work will be done on security issues related to exposing core network capabilities to the WWW. Furthermore the integration of an OMA compliant Policy Enforcer as part of OSE will be implemented to provide flexible mechanisms for Service Level Agreements (SLAs) between an operator and the service provider using the gateway.

References

1. BT. Web21C SDK. http://web21c.bt.com/.
2. Facebook. http://www.facebook.com/.
3. 3GPP. *TS 23.228. IP Multimedia Subsystem (IMS). Stage 2 v.7.10.0.* 2007.
4. H. Schulzrinne, et al.. IETF RFC 3261. *SIP: Session Initiation Protocol.* 2002.
5. P. Calhoun. IETF RFC3588. *Diameter Base Protocol.* 2003.
6. ETSI. http://www.etsi.org/tispan/.
7. Open Mobile Alliance (OMA). *Enabler Release Definition for Push-to-talk over Cellular. Candidate Version 2.0 – 11 Dec 2007.* 2007.
8. Open Mobile Alliance (OMA). *Presence SIMPLE Architecture Document. Approved Version 1.0.1 – 28 Nov 2006.* 2006.
9. Open Mobile Alliance (OMA). *XML Document Management Architecture. Candidate Version 2.0 – 24 Jul 2007.* 2007.
10. Open Mobile Alliance (OMA). *OMA Service Environment. Approved Version 1.0.4 – 01 Feb 2007.* 2007.
11. ECMAScript Language Specification, 3'rd Edition, 1999, http://www.ecma-international.org/publications/standards/Ecma-262.htm
12. J. J. Garrett. *Ajax: A new Approach to Web Applications.* 2005.
13. The XMLHttpRequest Object, W3C Working Draft, October, 2007 http://www.w3.org/TR/XMLHttpRequest/
14. D. Winer; XML-RPC Specification, June, 1999, http://www.xmlrpc.com/spec
15. JSON-RPC Specification 1.1, Working Draft, August, 2006, http://json-rpc.org/wd/JSON-RPC-1-1-WD-20060807.html
16. A. Russel, G. Wilkins, D. Davis, M. Nesbitt, Bayeux Protocol 1.0 draft 1, 2007, http://svn.xantus.org/shortbus/trunk/bayeux/bayeux.html
17. Google Maps API, http://code.google.com/apis/maps/
18. Yahoo Search API, http://developer.yahoo.com/search/web/
19. Youtube Data API, http://code.google.com/apis/youtube/overview.html
20. Facebook API, http://developers.facebook.com/
21. Programmableweb, http://www.programmableweb.com/
22. SIPGate API, http://www.sipgate.co.uk/user/download_api.php
23. Skype4Web, https://developer.skype.com/Docs/Web
24. FOKUS Open Communication Server, http://www.open-ims.org/ocs-x
25. Parlay X. http://www.parlay.org/en/specifications/pxws.asp, 2008
26. W3C, Extensible Markup Language (XML). http://www.w3.org/XML/
27. Apache Software Foundation. Apache Axis2. http://ws.apache.org/axis2/
28. Jabsorb Framework. http://jabsorb.org/
29. Pushlets Framework. http://www.pushlets.com/
30. Apache Software Foundation. Apache Tomcat Configuration Reference (Cross-Context). http://tomcat.apache.org/tomcat-6.0-doc/config/context.html
31. Google Inc. OpenSocial API. http://code.google.com/apis/opensocial/
32. A. Motanga, A. Bachmann, T. Magedanz, *Requirements for an Extendible IMS Client Framework*, Mobilware'08, February 12-15, 2008, Innsbruck, Austria, ACM 978-1-59593-984-5/08/02
33. Open SOA Telco Playground, http://www.opensoaplayground.org