

Context-Aware Content Casting

Telma Mota¹, Massimo Valla², Diogo Gomes³

¹Portugal Telecom Inovação, Aveiro, Portugal; ²Telecom Italia LAB, Turin, Italy; ³Instituto de Telecomunicações, Aveiro, Portugal;

E-mail: ¹telma@ptinovacao.pt, ²massimo.valla@telecomitalia.it, ³dgomes@av.it.pt

Abstract: This paper presents a context-aware content distribution architecture based on both a generic context framework and a service platform composed of smart service enablers which are reused by different applications. A general description of the architecture components is presented emphasising the context framework, its main features and functionalities. To prove the proposed concepts and evaluate the architecture three applications have been developed targeting different types of content and usage scenarios in Public Smart Spaces.

Keywords: context-awareness, smart spaces, content casting, pervasiveness, personalisation, adaptation

1 INTRODUCTION

Context-awareness can be defined as the capability of a system or a device being aware of its environment or situation and responding proactively, meaning adapting, based on such awareness [1][2]. In spite of being a concept existing for more than 10 years, the ubiquity of mobile devices and the proliferation of sensor technologies make currently context-awareness much more attractive for service developers. Artefacts and mobile devices can be everywhere, from clothing to buildings, and may acquire and transport valuable context information as well as actuate on the environment accordingly. Context-awareness is therefore the key feature for effectively building pervasive systems which main purpose is to reduce human intervention by increasing software intelligence. In order to achieve such pervasiveness, context information must be collected, transported, filtered, stored and represented in a proper and extensible way. In addition it is also important to infer, reason and predict more complex context information (e.g. situations) in order to effectively build context-awareness into applications and smart environments [3].

This paper is focussed on context-aware content casting; context information is used to intelligently personalise, select, rate, distribute and adapt content to user's needs. The proposed context-aware architecture built using a generic context management framework can be integrated in an operator Service Delivery Platform (SDP). It integrates smart service enablers for finding and grouping users with common interests, select the most appropriate content for each group and setup the best delivery mechanisms. This architecture is explained in Section 2 and relies on a common and integrated context framework presented in Section 3. In order to evaluate and better

present the potential of the proposed architecture some context-aware applications have been developed mainly for Android [4] phones but also for iPhones [5]. A testbed was setup to serve development and demonstration purposes. This work is being trialled and tested on the field, with real users that interact with the developed applications and platform. A preliminary user feedback is presented in Section 4.

The presented architecture is based on the work done in the European research project "C-CAST" (Content Casting), which aims at providing an end-to-end system design for context based content casting.

2 CONTEXT-AWARE CONTENT DISTRIBUTION ARCHITECTURE

Context-awareness is the most important "natural" feature in humans and the "wish to have" feature in software. Designing software similar to human behaviour has always been a big and interesting challenge for software developers [6].

The architecture proposed in this paper includes a set of pervasive service enablers, which can intelligently take proactive actions and perform automatic adaptations on behalf of the user. Although the focus of the proposed solution is personalised content casting, the architecture is flexible and extendable enough to easily accommodate numerous services. Different types of content (e.g. news, advertisements, user generated content – photos and videos) are recommended, rated and selected according to the user's situation, preferences and needs, therefore according to user's *context*. In our daily life, users share occasionally the same situation and interests with other users depending on several factors (e.g. the location, the place, the status, the type of event, etc). Such situations constitute opportunities for the delivery of content through efficient mechanisms that target groups instead of single users (e.g. 3GPP/MBMS [7]). Therefore, besides personal information (e.g. user profile, preferences, presence status), the context which better determines a personalised "good" and "appropriate" content delivery depends on the surrounding environment (e.g. location, sensors) and also on the available network conditions and the user's terminal(s) capabilities. Having access to this integrated global view in a common context framework is then crucial to applications and smart service enablers which will proactively identify those common conditions and group the users accordingly, in order to optimise content selection and distribution. At the session and network level some context-aware automatic intelligent actions may also be performed such as redirecting a

session when a new and more sophisticated terminal is detected (e.g. big panel), mute the audio stream when the environment is noisy and introduce subtitles, choose the best access network to transmit the content (e.g. wifi, 3G) according the signal strength and user preferences, select the best path (in case of a multicast transmission), allocate the right resources and seamlessly handover the stream when the current signal is too weak [8]. Given the need of proactively performing these different types of adaptations (i.e. grouping users, selecting content, controlling the session and manage the network) the proposed architecture (see Figure 1 1) includes four service enablers, one per type of adaptation: Group Management Enabler (GME), Content Selection Enabler (CSE), Session Management Enabler (SME) and Network Management Enabler (NME). These context-aware enablers interact with a global content framework in order to retrieve and insert context information. A Context Broker (CxB) (described in Section 3) is the main component of this framework and provides an interface to access a diverse set of context providers.

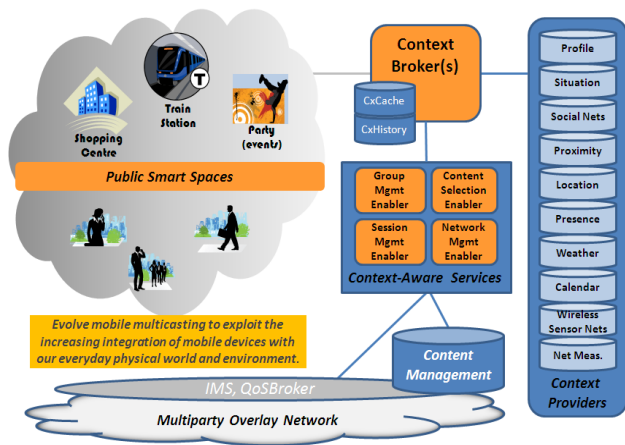


Figure 1: Context-Aware Content Casting Architecture

2.1 Group Management

The Group Manager Enabler (GME) is responsible for identifying, creating and managing user groups, based on their contextual information and group creation criteria specified by service providers. The group creation settings are collectively referred to as group definition. A group definition comprises of parameters like context scope(s) to monitor, minimum group size, and notification mechanism amongst others. The user context is retrieved from the Context Broker and processed by the GME to recognise group occurrences. The groups hence created are communicated to the respective application and to any other component (e.g. service enablers) which subscribes to group formation events. The GME acts both as a context consumer and as a context provider; subscribes to context updates to build groups and the resulting information (i.e. group Id, members of the group, criteria) becomes also available as part of the global context framework.

2.2 Content Selection

Selecting the most appropriate content to transmit is the key feature in context-aware content delivery systems. As context encompasses a large diversity of information, the selection process can span from trivial choices, like selecting content with the most appropriate language, to complex reasoning, like deciding that the content item most suitable to be sent is “traffic information” due to the situation “the user is travelling by car”. The Content Selection Enabler (CSE) identifies the proper content for a user or a group of users based on service configuration (criteria defined by the service provider), on context information and on available content and its metadata. The returned information is the content identification; it just refers to the actual content item independently from its format, resolution, or any other content property. The CSE may also be used as a recommender since it estimates content item ratings, always based on their context.

2.3 Session Management

The Session Management Enabler (SME) handles context-aware session control of multiparty sessions. It controls the session initiation, modification/renewal and termination. This service enabler interfaces with a “legacy” 3GPP/IMS platform in order to establish SIP sessions for the group of users selected by the GME. Note that in our target service scenarios (presented in section 4.1), sessions are server side initiated using a back-to-back user agent, given that it is up to our platform to decide on behalf of the user (depending on context information) what is the most appropriate content to distribute, when is the best occasion to do it and how (unicast, multicast, WiFi, 3G, etc.). Therefore, the typical SIP negotiation process may be simplified; the SME is aware of the terminal capabilities and network conditions in advance through the Context Broker and is also informed about the content description that can automatically be adapted to those conditions. This enabler adds context-awareness to the “legacy” 3GPP/IMS CSCF (Call Session Control Function) [9]; session modifications are triggered by context changes.

2.4 Network Management

Before a session is set-up a request is sent by the SME to the Network Management Enabler (NME) to check the available network conditions, user’s terminal capabilities and user preferences; the NME will create subgroups according to different QoS needs, make the proper resource reservations and select the right path (in case of multicast). It will also automatically reconfigure in case of any network change and will trigger seamless handover (network initiated) if needed. Similarly to the SME this component introduces context-awareness in network management and interacts with a “legacy” TISPA/RACS (Resource Admission Control System) [10].

2.5 Content Management

The spread of heterogeneous, video-enabled devices such as computers, mobile phones, and PDAs has triggered the need for more dynamic content adaptation to users' conditions and needs. The purpose of the Content Management module (which includes Content Processing and Delivery) is therefore to provide the final and best suited personalized content to the end-user. The Content Management Module is composed of a metadata storage function in which content files are automatically associated to metadata and context information and of a content processing and distribution function through which content is adapted and streamed. The content processing function provides the content based on the CSE, which matches the group identifiers provided by the GME. Content distribution function implements a SIP UA interface with the SME and can stream content using the RTP protocol.

2.6 High-Level Functional Description

Figure 2 presents a high-level functional description of the enabling architecture. The process starts when each application configures criteria in GME and CSE (rules for grouping and for content selection – application specific).

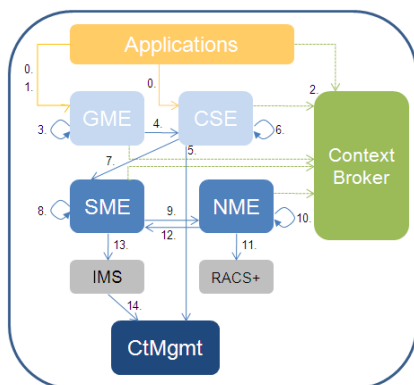


Figure 2: High-Level Functional Description

1. Then the GME should receive the list of users to be monitored and/or get subscribed users from the CxB.
2. All context-aware smart service enablers must subscribe to the CxB in order to be notified of context updates. Also applications may subscribe to show user related context info (e.g. name, age, mood, preferences, temperature, etc.).
3. The GME will start creating groups of users as soon as the initial configured conditions based on context are met for a specific application (e.g. group all children located in Berlin who are on line and like Simpsons).
4. After subscribing, CSE gets notified of group creation and/or of group changes.
5. CSE gets also content metadata every time groups are created/modified or periodically.
6. CSE selects and rates content based on context and metadata which better meets the criteria initially configured by a certain application. Based on weights

the CSE decides which content better matches the defined rules.

7. CSE triggers then session set-up for the first content of the rated list (only valid for streaming and downloads).
8. SME establishes sessions according to terminal caps (e.g. codecs), and user preferences. Session modifications (e.g. mute, delete or add a flow) are also triggered by context changes.
9. SME triggers then resource reservations in NME.
10. NME creates subgroups according to terminal caps, user preferences (e.g. wifi or 3G), content description (e.g. bit rate) and network measurements (e.g. delay, jitter and loss). Also creates multiparty communication paths.
11. NME requests enforcement for the best quality possible on top of the decided path and activates overlay nodes if needed.
12. NME confirms or refuses the SME request
13. In case of confirmation from the NME, the SME requests SIP session set-up.
14. And finally Content Management is triggered (e.g. Streamer) accordingly by the IMS platform.

This high level description shows how the proposed architecture enables personalising of multicast-broadcast content delivery, despite of the apparent mutual exclusivity of the two, and facilitates the realisation of add-value benefits for the service, content and network providers as well as for an eventual future new business roles: context broker and context providers.

3 GLOBAL CONTEXT FRAMEWORK

Context Information is managed by the modular and integrated Context Framework described in this section.

3.1 Framework Roles and Components

The core of this framework is the *Context Broker* that relies on several *Context Providers*; *Context Consumers* can access these components in order to retrieve context information and *Context Sources* (typically located on mobile devices) can provide context data to the Context Broker.

3.1.1 Context Broker

The Context Broker (CxB) works as both handler and aggregator of context data and also as an interface between architecture components. Primarily the CxB has to control the context flow among all attached components and has to “know” all Context Providers in the architecture; this is realised through an advertising or registration process by Context Providers.

3.1.2 Context Provider

A Context Provider (CxP) is a module that provides context information in synchronous mode; a Context Consumer or even the Context Broker can invoke the CxP in order to acquire context information. Moreover, a CxP can produce new context information inferred from the computation of input data. Every CxP registers its

availability and capabilities by sending appropriate advertisement to the CxB and exposes interfaces to provide context information to the CxB and to Context Consumers. In order to have a good “picture” of the activities of the user and the surrounding environment the implemented context providers include heterogeneous and complementary types of information (e.g. location – operator, preferences – Internet Facebook, Presence – xmpp server, sensors, network QoS measurements, situation – inferred information)

3.1.3 Context Source

A Context Source (CxS) updates context information, about one or more context domains, in asynchronous mode. A CxS sends context information according to its internal logic and never due to an invocation from another middleware component. Typical context sources are the terminal agents that due to quering limitations from the server side produce information cyclically.

3.1.4 Context Consumer

A Context Consumer (CxC) is a component that gets context data. A CxC can retrieve context information by either sending a request to the CxB or invoking directly a CxP over a specific interface. As an alternative, the CxC can obtain information using implicit request to the CxB, meaning by subscribing to a specific context update event and being notified by the CxB when events occur. Context consumers are typically context-aware components, as user applications, smart service enablers (as those described in the previous section), and context providers that play more than one role: consume, infer and provide the resulting context information to be used by other components.

3.1.5 Entity

Exchange of context data always refers to a specific *entity*. An entity is therefore the subject (e.g. user or group of users), to which context data refers to, and it is composed of two parts: a *type* and an *identifier*. Every CxP supports one or more entity types. This information is published to the CxB during the advertising process. An entity type categorises a set of entities; example of entity types are: *username*, *imei* (device), *SIP uri* (service) and *groupid* (group).

The entity identifier specifies a particular item in a set of entities belonging to the same type. For example, entity *username/mrjones* specifies the *mrjones* user in the platform. Also every user entity in the platform can be related to his devices and service accounts entities; by means of an *entity resolution* mechanism the Context Broker is able to return context for a user event if context was generated for one of his devices. Considering for example a CxP that provides geographical cell-id based location for mobile devices; if the location information is obtained from the computation of parameters provided by mobile devices, this CxP supports entity type *imei*, and the Context Broker will be able to return location for the owner’s user entity.

3.2 Features and Functionalities

3.2.1 Context Scopes

Context information is grouped in *scopes*, which are a set of closely related context parameters. Every context parameter has a name and belongs to only one scope. Using scope as context exchange unit is very useful because parameters in that scope are always requested, updated, provided and stored at the same time (i.e. the creation and update within a scope are always atomic and that context parameters in a scope are always consistent). Scopes themselves can be atomic or aggregated, as union of different atomic context scopes. For example, the scope *position* refers to the geographic position in which an entity is: this scope is composed of parameters *latitude*, *longitude* and *accuracy* (error on location) and these values are always handled at the same time.

3.2.2 Context Brokering

Context brokering is a main feature of the context framework. As shown in Figure 3 the Context Broker is responsible for retrieving context information from Context Providers and storing information received from Context Sources.

The main tasks for the CxB are:

- Enable the discovery of Context Providers;
- Manage the exchange of context information between providers/sources and consumers;
- Caching context data by mechanisms of storing, retrieving, querying, deleting, and matching context elements.

A Context History is maintained in a central database.

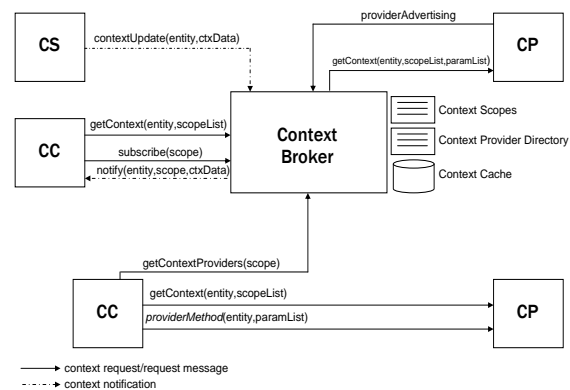


Figure 3: Context Framework

CxPs register to the CxB (*providerAdvertising* method) by sending both the list of scopes it can provide and the list of parameters needed for its invocation (URL, other context parameters belonging to different scopes).

CxSs push context information asynchronously to the CxB (*contextUpdate* method), containing both the list of the scopes with their names and parameters and the entity which they refer to. The CxB saves this information in its cache.

CxCs can obtain context information in the following three ways:

- Explicit query to CxB: CxC directly requests context information, even if it is not aware of context provider references and provider methods. The CxB retrieves the context information, searching for a CxP providing the required scopes and invoking it;
- Explicit query to CxP: CxC directly requests context information to provider interface. CxC can previously request CxB for information about context provider scopes and provider references (*getContextProviders* method);
- Event-driven (subscription to CxB): CxC subscribes to context change events (e.g. *onContextChange* of any scope, i.e. *subscribe* method) and is informed when they occur, by means of a call-back to a specified URL.

In the last case, applications interested in being notified about changes of context may subscribe to such event. The CxB infrastructure is thus, responsible for pooling the current context and delivering such information to the context consumers who have subscribed to the related context changes.

3.2.3 Context Caching and History

Context information (scope) received by the Context Broker (from a CxS or as a result of a request to a CxP) is stored in a *Context Cache*. If another CxC requests the same scope to the Context Broker, it can be retrieved from the cache, if it is not expired, without need to invoke the same Context Provider again and therefore speeding up the process of context delivery.

Every context scope exchanged between the Context Broker and Context Providers or Context Sources is also logged in a *Context History*. Differently from the context cache, which stores only currently valid information, context history can provide the past context information of an entity. Data mining and context reasoning techniques can be applied in order to correlate context data and deduce further context information.

3.3 Interfaces and ContextML

In the proposed architecture, all components which provide context information expose REST-like interfaces over HTTP protocol, allowing easy access by simply invoking a specific URL. The common interface URL structure is:

```
http://<SERVER>/<CONTEXT_MODULE>/<INTERFACE>/<OPERATION>/<RESOURCE>? [<OTHER_PARAMETERS>]
```

In order to allow distribution of heterogeneous information, context data needs to be enclosed in a common format understood by the CxB and all other architectural components. This format is called *ContextML* (*Context Markup Language*).

3.3.1 ContextML

ContextML [11] is an XML-based language for context representation and communication between components; it also provides commands to enable CxP to register

towards the Context Broker and enables Context Consumers to discover the context information they need. ContextML provides the following features: representation of context data, advertising of Context Providers, description of Context Providers published to the Context Broker, description of context scopes available in the Context Broker and representation of generic responses (ACK/NACK).

4 TESTBED AND USE CASES

In order to evaluate and test the aforementioned architecture a distributed testbed was built. The core of the tested is currently deployed in a distributed way, in Portugal, Italy and Israel, but can be accessible worldwide as all services are made available through public addresses.

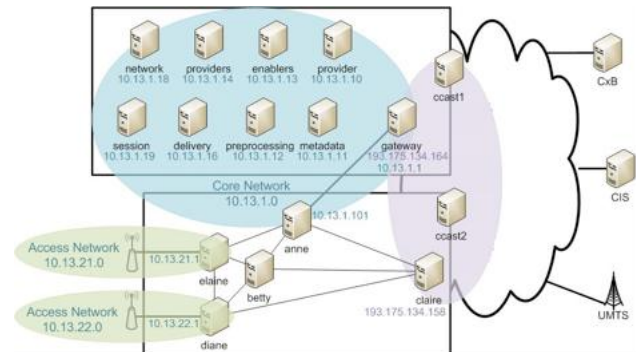


Figure 4: Testbed

The testbed is physically composed of two Virtual Machine (VM) Hosts (C-Cast1 and C-Cast2) running VMWARE ESXi 4.0 [13]. C-Cast1 hosts the Application Servers in which the Context Providers and Service Enablers are installed while C-CAST 2 hosts 5 routers connected (Figure 4). Two WiFi access points are connected to C-Cast2 as well as a commercial 3G network accessible through one of the virtual routers. These 3 routers provide real wireless connectivity to the virtual network. Besides the components deployed in the VM, there are two components that are deployed in the premises of the partners responsible for each of them: the CxB (Context Broker) in TI-Labs, Torino, Italy and the CIS (Context Inference Server) used by the GME in IBM, Haifa, Israel. Several Android Smart phones and some iPhones exist both in the same premises of the testbed, as in partners' locations around Europe and provide the main vehicle for the testing and demonstration of the developed architecture.

4.1 Use Cases

The supporting European project (FP7 C-CAST) considered, through the course of this research, 3 main smart-environment use-case scenarios: Mass Transportation (e.g. Trains), Shopping Centres and Social Gatherings (e.g. parties, music festivals). The definition of the 3 use-case scenarios lead to the development of 3 applications that relied on the proposed architecture enablers to achieve its goals (i.e. Train App, Mall App and Party App). The applications were developed using a

Client/Server paradigm in which the Client was implemented in the Android Platform and the Server in a Web Application Platform. All client applications share a common code base consisting of a SIP/IMS stack for signalling purposes, a Webkit [12] component for interaction with the server side applications part and the Android native media stack. In the Server side, applications were mainly composed by the interfaces to the enablers and lightweight set of rules that controlled the possible interactions between service enablers.

In spite of all applications being targeted to Public Smart Spaces, each of them intends to test and highlight a different aspect of the proposed architecture.

The first application (Train App) targets large and broad sets of potential users geographically disperse that are travelling a railway network. The application provides its users with professional content based on the users' time availability and preferences according to their social profiles (extracted from a social network). The application can dynamically react to changes in user context such as train delays and provide contextualized information to users on how they should precede to destination.

The second application (Mall Application) targets visitors of commercial district buildings in which shopkeepers have set forward marketing campaigns to attract customers based on their proximity and preferences. Target video advertisements are one of such examples, and each user is able to get more personalised content.

The last application (Party Application) focus is on User Generated Content (UGC). This application offers the possibility to gather users in ad-hoc events (e.g. party) and associate the context of that event to the content produced at the same event. Content produced through this application is embedded with additional context information that makes it more relevant for future reference.

4.2 Field Trial

In addition to the setup of the testbed consisting of the architectural components and applications, a field trial is being conducted in order to evaluate Usability and User eXperience (UX) in a pilot setting, with students of the University of Aveiro, Portugal in an environment that simulates a real situation. The initial field trial is being conducted with evaluators who are available to participate and answer to a pre-questionnaire. The pre-questionnaire purpose is to identify the technological literacy, usage of social tools habits and consumption of content habits, as well as to form groups with different characteristics (age / literacy / consumption habits / etc) that nonetheless may share interests and therefore profit from grouping and context-awareness. The evaluators are being divided in four groups, who experience the application in different terminals: HTC Hero, Google Dev Phone Google Nexus and HTC Desire. A monitor accompanies the users and takes notes about the evaluator's behaviours and comments. The field trial is being conducted through a predefined path around a commercial area in which the evaluator's context changes and content is selected based

on the new context. At the end of the path they are faced with yet another questionnaire in which they can classify and comment their experience with the concept of public smart spaces and context based selection of content. Early results of the field trial pointed towards an inadequate length of the videos with 46,7% of users stating that the videos were too long. And to the identification of noise and attention dispersion as the main factors that compromised user experience (>60% answers). Nonetheless evaluators rated the concept positively with 57,4% considering the application easy to use. Remarks were made about privacy (88%) and need to personalize the levels of importance given to the context (74%) (e.g. suggestions were made that content based on location might not be important for some users).

5 CONCLUSIONS AND FUTURE WORK

This paper presented an end-to-end context-aware content distribution architecture based on a generic Context Framework and implemented on top of existing technologies such as those defined in IMS and TISPAN. The proposed architecture has been successfully implemented and three different applications have been built based on this platform in order to better evaluate context-awareness in Public Smart Spaces, paving the way to future smart cities

An initial and limited field trial has already indicated directions in which further research needs to be applied in the areas of Privacy and User Experience.

Acknowledgment

The authors would like to thank all partners who participated in the FP7 C-CAST project (<http://www.ict-cast.eu/>) and contributed to the results presented in this paper.

References

- [1] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, pp. 4–7, 2001.
- [2] N. Baker, et al, "Context-Aware Systems and Implications for Future Internet" *Future Internet Conference*, Prague, Czech Republic, May 2009
- [3] B. Moltchanov, et al, "Context Management and Reasoning for Adaptive Service Provisioning", *International Conference on Ultramodern Telecommunications*, St. Petersburg, Russia, October 2009.
- [4] <http://www.android.com/> (accessed July 2010)
- [5] <http://www.apple.com/iphone/>
- [6] A. K. Dey , G. D. Abowd, "Towards a better understanding of context and context-awareness " *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999
- [7] 3GPP TS 23.246, available <http://www.3gpp.org/ftp/Specs/html-info/23246.htm>.
- [8] Simoes, J., et al. "Context-aware control for personalized multiparty sessions in mobile multihomed systems". *MobiMedia 2009*, Brussels, Belgium
- [9] <http://www.3gpp.org/article/ims>
- [10] Neto, A, et al. "Multiparty Session and Network Resource Control in the Context Casting (C-CAST) Project", *Lecture Notes in Computer Science*, ISSN: 0302-9743.
- [11] Knappmeyer M. et al. "ContextML: A light-weight context representation and context management schema", *Wireless Pervasive Computing (ISWPC)*, 2010 5th IEEE International Symposium on, pp. 367-372, 5-7 May 2010, Italy - doi:10.1109/ISWPC.2010.5483753.
- [12] <http://webkit.org/> (accessed July 2010)
- [13] <http://www.vmware.com/products/vsphere-hypervisor/> (accessed July 2010).