# A comprehensive framework for context-aware communication services

Bachir Chihani[1,2], Emmanuel Bertin[1]
[1] Orange Labs
42, rue des Coutures
14066 Caen, France
{bachir.chihani,emmanuel.bertin}@orange-ftgroup.com

Noel Crespi[2]
[2] Institut Telecom, Telecom SudParis, CNRS 5157
9 rue Charles Fourier
91011 Evry, France
noel.crespi@it-sudparis.eu

***Abstract*— Context-Aware Communication (CAC) services enable end-devices or service platforms to adapt their comportment to sensed context information, either instantly or in a deferred way. There is today no available framework for supporting the development of CAC applications in a generic way. After surveying previous works, we propose here a new approach for classifying context-aware communication systems. Relying on this approach, we design a comprehensive framework based on a distributed service broker for the development of such systems. We validate this framework by implementing a prototype and by testing it.**

***Keywords – Ubiquitous Systems; Context related services; Context-awareness; Communication services***

## I. INTRODUCTION

Context-Aware Communication (CAC) services are applications that rely on current and previous user's context to facilitate user's communications.

The most used context sources in CAC solutions are physical information (e.g. location and time), environmental information (e.g. weather, lighting, signal strength and sound), personal information (e.g. agenda, health, and mood), social information (e.g. relationship between people), and applicative information (e.g. sent and received e-mails). High level knowledge can be derived from this raw contextual information to draw a better understanding of the user, like for instance predicting the most appropriate time for the caller to send a communication request to the callee.

Previous works for classifying CAC focused either on the system's autonomy level of acquiring context information and for using it to perform actions [1], or on the operation modes [2]. These approaches classify CAC based on the goals of the application or on the kind of service it delivers (for example routing calls to the most appropriate destination). We propose here another approach for classifying these solutions based on where the adaptation is performed and on how the context is used. This segmentation provides a way to classify CAC systems based on their implementation instead of their functionalities. It enables us then to build an underlying framework for every kind of CAC services.

In the rest of the paper, we first survey the existing works on CAC services, both from academia and industry, and we propose a segmentation to classify them. We then introduce a comprehensive framework to realize these CAC services. This framework is based on a distributed service broker. Finally, we detail the implementation issues and the testing of our prototype with a typical use case.

## II. A SEGMENTATION OF CAC SERVICES

Our approach for classifying CAC is based on two axis, as shown on figure 1:

1. How the sensed contextual information is used, either instantly or in a deferred way, for example after processing a huge amount of sensed data with data mining techniques;
2. Where the adaptation is performed, either on a service platform or on an end-user device.

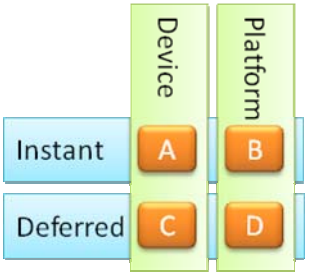


**Figure 1: CAC segmentation**

### A. *Case A*

This case gathers the solutions that exploit instantly the sensed context to perform the adaptation on the end-user device. An example of service from this class is a Rendez-Vous service that aims to help a group of people to organize a meeting by sensing the calendar data of each one to check their availability in the future.

It includes also CAC solutions for smartphones that makes callee and caller more aware about each other while they are in communication by enabling the exchange of information

about the way they are holding the phone and reproducing it at each side in a non intrusive and smooth way.

In [8] the authors present the Citron framework for acquiring and processing contextual information from personal devices. They use a prototype of a user device, called Muffin embedded with multiple sensors classified into environmental sensors (e.g. air temperature sensor, relative humidity sensor and barometer), biological sensors (e.g. alcohol gas sensor, pulse sensor, skin temperature sensor, skin resistance sensor), Motion/Location sensors (compass/tilt sensor, 3D linear accelerometer, grip sensor, ultrasonic range finder, GPS), and other sensors (e.g. RFID sensor, front/rear camera and microphone). The acquired context is used to track the user's state and path, and then display the pathway in the device's screen.In [11], the authors present ConChat, a context-aware chat application that allow users to exchange contextual information (e.g. location, mood). Users can also save their privacy by restricting the access to their contextual information to only allowed persons.

In [15], the authors present Calls.calm a prototype for enabling caller to be aware about callee's situation which is manually updated and to decide if it is suitable to initiate a communication at a certain time. When the caller intends to contact the callee, he will be redirected to a web page containing a personalized greeting from callee, information about his situation, information about interaction between callee and caller (e.g. previous messages), and a set of communication channels (e.g. direct call, text and voice message).

### *B. Case B*

This case gathers the solutions that exploit instantly the sensed context to perform adaptation at the platform level. An example of such solutions is location based messaging like Socialight [3] a research project that becomes a commercial service. The aim is to allow users to post messages so that it will be delivered to anyone who is located nearby.

In [10], the authors propose a context-aware communication assistant named INCA that uses callee preferences and context to handle communications (e.g. forwarding calls). This assistant relies on a layered approach in order to support adaptive context-aware communications. The main layers are session layer for signaling protocols, communication layer that include the environment of the communication (e.g. callee/caller identifiers, networks and devices), and user layer that models users' preferences and profiles to provide personalized services. Each layer (e.g. communication) holds an internal state which is updated after the reception of events from the lower layer (e.g. session) and the execution of actions requested by the upper layer (e.g. user).

In [11], the authors present Mercury, a SIP-based context-aware communication system that is able to support communications initiated from any device (on which the Mercury client is installed) and to route the communication based on the callee/caller preferences and context (e.g. device).

In [16], the authors present iCAMS, a prototype of a web application accessed from PDA that enable caller and callee to be aware of each other to facilitate communication in mobile environments. It uses location information through the PHS (Personal Handyphone System) cell phone network, and user's meeting information inputted manually. These contextual information are then used at the server side to sort the communication channels through which the user can be reached. For example, when the user is at work then the office number will be listed first, if he's in a meeting then the office email address is listed first, if he is at home then the house phone number will be listed first. The user can specify his owns rules to personalize the way communication channels are sorted. When the caller want to contact the callee, he access the web page corresponding to the callee to find the different communication channels sorted according to callee's situation and preferences. In case of emergency, the caller can ignore the way channels are sorted and use the best mean he thinks is suitable.

### *C. Case C*

This case exploits in a deferred way the sensed context to adapt the interface on the end-user's device. A typical example is the adaptive address book in which contacts within a contact list are sorted based on their relative importance (e.g. professional contacts shown first during working hours). Another example is context-aware recommendation system [4] that aims to use contextual information for making recommendation more relevant.

In [9] the authors propose an assistive system for disabled users who are limited in mobility and have loss of speech. The framework they developed aimed to support interactive communication for disabled people by predicting conversions based on historical data (e.g. discussions topics and phrases). It creates a distinct profile for the disabled user and his visitors. The visitors are classified as familiar and unfamiliar, and accordingly to their relationship with him (family, friends, healthcare professional).

In [14], the authors present BayesPhone, a prototype of a mobile application for handling incoming calls during a meeting by comparing the cost of interruption caused by the call and the cost of deferring it. From meeting's information available on the user calendar, the prototype builds a user model for interruptability and meeting attendancy. It also uses real-time sensed data about caller (e.g. his importance to callee) and callee (e.g. motion, conversations).

In [17], the authors present a context-aware mobility management system for mobile phones. The system is able to manage the connectivity of mobile applications in a transparent way by deciding which interface to use at a certain time and performing horizontal handovers between two network access of different technologies (e.g. WiFi and WiMAX). It can use different contextual information like network interface metrics (e.g. received signal strength), device characteristics (e.g. battery, memory), GPS coordinates, etc. The system uses statistical machine learning techniques to generate from sensed contextual information predictions about availability of network interfaces, and then it

decides when performing handovers. Predictions are made possible due to temporal and spatial patterns of user's daily behavior (e.g. taking the same way to go work every morning).

### D. Case D

This case is about the use of sensed context in a deferred way at the platform level. Works performed by N. Blum et *al.* [5] or K. Hamadache et *al*. [6] are good illustrations. These works share indeed the functionality of routing incoming communication based on the callee context (including the importance of the caller to the callee) and on a set of predefined IF-THEN rules (e.g. if the callee is busy and if the communication is not urgent, then redirect the caller to the callee's voice box).

Recently, few works have been conducted to extract new knowledge from data acquired from user's mobile devices. Reality Mining study [13] is an example of such works, it aims to recognize social patterns by analyzing a dataset of phone's contextual information of 100 students stored during the academic year. In [18], the authors use the data set (location and Bluetooth data) made available by the MIT's Reality Mining project to perform a study of human activities and recognize recurrent patterns. The recognition of recurrent patterns is with big benefit because it makes possible the automation of certain actions.

## III. RELATED WORKS OR EXISTING FRAMEWORKS

A lot of frameworks have been proposed for developing context-aware systems. In this section we will try to present some of these frameworks, and study whether or not they address the cases presented previously.

The design of the Citron [8] framework is based on a blackboard model. It is composed of a set of workers in charge of retrieving context from sensors or processing the stored context, and a shared space for storing context and sharing it among workers in a loose coupling way. Citron is to be implemented on devices but do not support communication between different instances installed on different devices. It supports the development of context-aware applications of class A as presented previously. But it does not provide an explicit management of historical data (e.g. use of timestamps) and thus it's hardly usable for building applications of class C.

In [12], the authors present ContextPhone, a platform for smartphones using Symbian OS (e.g. Nokia Series 60) above which developers can build adaptive applications that make use of contextual information available on the mobile (e.g. location, phone alarm profile, information about calls, surrounding Bluetooth devices). ContextPhone is to be installed on user's devices, it disposes of a communication module that enables sharing of contextual information, and it also supports the storage of sensed context. But it does not provide an explicit way (e.g. reasoning) to process historical data. Thus it supports only the development of context-aware applications of class A. The data set of devices' contextual information build from the logs of this platform was used in many studies, like the Reality Mining [13].

The framework proposed in [6] is composed of Context Source insuring the context management, and which is composed of a context wrapper and a reasoning engine. The wrapper has a module responsible of retrieving raw data from log file on the Telephony Application Server and updating the ontology with the aggregated data; Context Broker(s) to detect the need of an application to adapt to its context; User interface to enable the user to monitor his context and to setup his preferences, and also to enable the system configuration; Context Provider to ensure the communication between the source, the broker and the user interface. The framework is to be installed on a server and only a GUI is provided to the user to control his context and edit his preferences, also it supports the management of historical data (e.g. call logs). Thus, the framework supports only context-aware applications of class B and D.

The authors in [19] present Omnipresent, a service-oriented architecture for location-based applications. The proposed services within this architecture are a location-based service (e.g. maps, navigation, and advertisement) and a reminder service. Example of the first class of services is the delivery of maps enhanced with point of interest (POI) information. The aim of these services is to use contextual information retrieved from the user device (e.g. GPS location) to help him finding products, services, POIs, friends and management of their daily tasks. Omnipresent services are supposed to be installed on the server side where is performed the whole intelligence. Thus, most applications we can develop with Omnipresent are of class B or D.

Vimoware [20] is a middleware-based toolkit that supports the development of web services on mobile devices for collaborative purposes. The advertisement and discovery of services are implemented in a P2P-based subscription/notification approach which is suitable for ad hoc environments. Vimoware supports the hosting and advertising of SOAP-based web services, sharing of multimedia documents through HTTP, the provisioning of contextual information about devices (e.g. network, CPU, memory) and users (e.g. name, skills), management of users and teams, the management of flows of tasks (creation, control and execution), the support of communication (e.g. IM). The contextual information can be accessed by client applications (i.e. context-aware applications) directly via XPath/XQuery requests or by subscription. The middleware can be installed on different devices; it is able to make them communicate but doesn't provide an explicit management of historical data. Thus, Vimoware can support only context-aware applications of class A and B.

inContext [21] proposes an advanced SOA-based collaborative working environment for emerging team forms, and context/interaction based techniques for adaptation in collaborative environments. inContext manages different contextual information about users, services, teams, and activities. It is able to extract and analyze interactions between team members and infer knwoledge with interaction mining techniques. inContext uses sensed context and infered knowledge to support the adaptation which consist of the

contextual selection of collaboration Services: documents sharing (Document management and search), communication (SMS, IM, Email), team and project management (User, Activity and Team Management). inContext architecture is centralized where the intelligence is implemented at the server side. Thus, it supports context-aware applications of class B and D.

Context Broker Architecture (CoBrA) [22] is a broker-centric architecture for supporting context-aware systems. It is composed of a central agent named the context broker in charge of managing contextual information, and a set of other agents that are in charge of either the provisioning of context or the execution of actions. It uses OWL (Web Ontology Language) ontologies as context modeling language, and a rule-based context reasoning. It also protects user's privacy by allowing them to define rules controlling how their contextual information is shared. The context broker is supposed to be at the server side. At the device side, agents can be installed to support the adaptation of local services. Thus, the CoBrA framework supports the development of any class of context-aware applications.

MUSIC [23] is middleware-based and a layered architecture helping the development of context-aware systems. It supports the management (collecting, organizing, and sharing) of context and its distribution among different instances; also it supports the adaption of applications according to context changes. It provides administrators with tools to manage it and optimizing resource usage. The MUSIC middleware runs on Java Virtual Machine (JVM), and can be installed on any device (e.g. PC, mobile). It can be used for developing any class of context-aware applications.

The framework presented in [17] is to be installed on mobile devices but does not provide support for communication between different instances. It supports the management of historical data and its combination with sensed context as illustrated in section II.C. Thus, the framework can be used for developing context-aware applications of class A and C.

We summarize the classification of the different frameworks in the following table.

**Tableau 1: classification of related works**

| | A | B | C | D |
|---|---|---|---|---|
| Citron | + | | | |
| ContextPhone | + | | | |
| Framework[6] | | + | | + |
| Omnipresent | | + | | + |
| Vimoware | + | + | | |
| inContext | | + | | + |
| CoBrA | + | + | + | + |
| MUSIC | + | + | + | + |
| Framework[17] | + | | + | |

From this comparison we conclude that previous works mostly focus on either performing the adaptation at the server side or at the terminal side. We propose next a framework able to address adaptation at both sides.

## IV. A COMPREHENSIVE CAC FRAMEWORK

Relying on our segmentation approach and the analysis of existing frameworks, we have implemented a generic framework (figure 2) for developing any class of CAC systems. The framework is distinguishable from previous works by its clear separation of concerns (production, distribution and consumption of context), its scalability provided by the framework's hierarchical architecture, and the symmetry of components implemented at the device and the server.

This framework enables the adaptation on both device and platform sides (our second segmentation axis) by implementing the consumer/producer design pattern, with the introduction of a distributed broker. The producers represent the source of context, while the consumers represent the applications that consume context to adapt their behavior. The broker is used to decouple consumers (who subscribe for desired contexts) from producers (who publish context information). This broker is distributed between the device and the platform sides: context consumer and provider are seamlessly on the device or on the service platforms. Moreover, a context history module stores all the context information published by every context providers, whatever his location (device or platform side). This enables to log an historical view of the context in order to process it with data mining algorithms and then to use it in a deferred way (our first segmentation axis).

Communication between the various components of the framework is either based on synchronous (request\response) or asynchronous (publish\subscribe) communications. In the asynchronous mode, the consumer subscribes to a type of context information and is then notified by the broker when a provider publishes such information. In the synchronous mode, the consumer requests context information from the broker that responds immediately, as it is continually gathering information from all the providers in the context history module. The asynchronous mode is likely used for instant consumption of context information, while the synchronous mode is used for deferred consumption.

From the implementation viewpoint, there are two interrelated levels of brokers: a platform side broker and a device side broker. The former has knowledge about the context information from all providers and from the one produced after processing historical data. The later has knowledge about local context, and requests if needed more global information from the platform side broker.

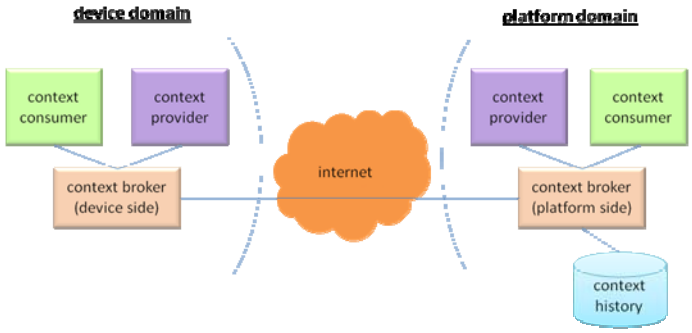

**Figure 2: A framework for CAC**

## V. IMPLEMENTATION AND TESTING

In the platform domain, the broker is developed in Java with the RESTlet framework [24] and exposes RESTful web services. Providers publish context events using a REST interface. Consumers subscribe to context information using another REST interfaces and are then notified with this information through a callback mechanism. Concerning the device domain, we implemented the device side of the broker on a smartphone and on a PC. On the mobile device, we developed our prototype with Android. The device-side broker relays when necessary the context subscriptions to the platform-side broker. It get notified through the Android C2DM (Cloud to Device Messaging) mechanism. The Personal Computer (PC) client is developed with Java and notifications are handled with a CometD server [7].

Next we will present how we have used our framework to implement a context-aware system.

### A. *Service presentation*

Nowadays we live a big enhancement in communication tools, and there are many ways that can be used by people to communicate, like email, phone, or Instant Messaging. This enhancement brings a certain amount of stress on people, especially for employees, because they are losing control on how and when they can be reached.

One of the possible solutions to alleviate this amount of stress is to delegate the control of interruptions to the user's contact by publishing his contextual information. The published information will help his contacts to evaluate the impact of the interruption caused by initiating a communication with him.

From this idea we derived a prototype that we called HEP (enHanced unintErruPtion) [25] with the aim to recommend the caller the best communication tool based on the callee context.

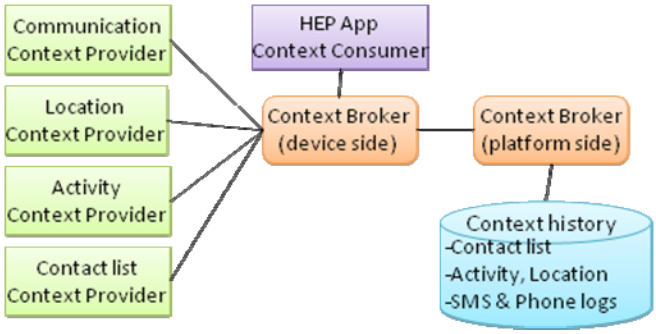


**Figure 3: HEP architecture**

### B. *Service implementation*

Figure 3 illustrates the architecture of the mobile application. The HEP App (figure 4) is an Android application enhancing the address book with contextual information about the user's contacts. The used contextual information are communication history (phone calls, and SMS), the Android address book from which we retrieve the user contact list, the user location and activity (the user is asked to set it manually).

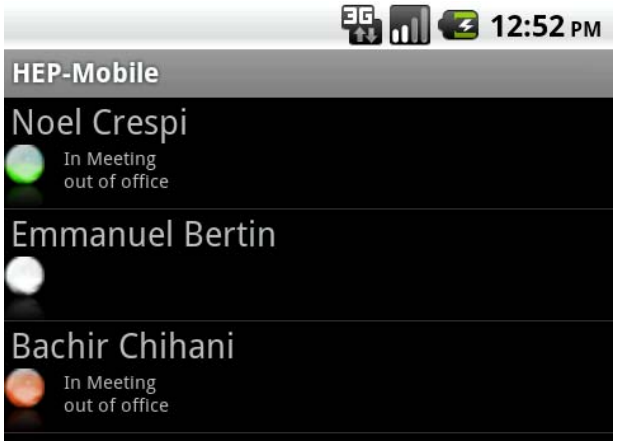


**Figure 4: A snapshot from HEP Android application**

From the extracted context we generate a workload status of the user that is then displayed to the user's peers in order to inform them about his ability to answer a call or read a received message. The different possible workload statuses are illustrated in figure 5.


**Figure 5: workload statuses**

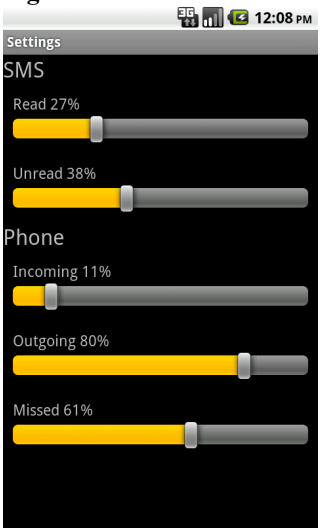


**Figure 6: Personalization of the weight values of the different contextual information**

The workload is calculated using the following equation where *context* corresponds to the sensed information; *weight* corresponds to the weight of the corresponding context in the calculation of workload:

$$workload = \frac{\sum_i context_i \times weight_i}{\sum_i context_i}$$

Users can customize how their workload is generating by personalizing the different *weight* values (figure 6).

## VI. CONCLUSION

Our approach for classifying CAC enables to quickly and precisely analyze and compare existing solutions. It thus helps to identify where further research is needed.

In addition, our framework constitutes a value-added infrastructure to build CAC services, relying on the proposed classification approach. As it is based on a distributed context broker, it handles in a transparent way the distribution between the device and the platform side, as well as the consumption of context information by the applications instantly or in a deferred way. At the moment, security and privacy issues are not addressed.

## REFERENCES


[1] B. Schilit, D. M. Hilbert, and J. Trevor, "Context-aware Communication," *IEEE Wireless Communications* 9(5), 2002, 46-54.

[2] F. Toutain, A. Bouabdallah, R. Zemek, C. Daloz, "Interpersonal Context-Aware Communication Services," IEEE Communications Magazine, January 2011.

[3] D. Melinger, K. Bonna, M. Sharon, and M. SantRam, "Socialight: A Mobile Social Networking System," New York University, 2004. http://www.socialight.com/

[4] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada, "Fusing Mobile, Sensor, and Social Data To Fully Enable Context-Aware Computing," HOTMOBILE, Annapolis, Maryland (USA) 2010.

[5] N. Blum, S. Lampe and T. Magedanz, "Enabling Context-Sensitive Communication Experiences," *14th International Conference on Intelligence in Next Generation Networks* (ICIN 2010), Berlin, Germany, October 11-14, 2010, ISBN 978-1-4244-7443-1.

[6] K. Hamadache, E. Bertin, A. Bouchacourt and I. Benyahia, "Context-aware communication services: an ontology based approach", 2nd International Conference on Digital Information Management (ICDIM'07). Lyon, France. October, 2007.

[7] CometD: http://cometd.org/

[8] T. Yamabe, A. Takagi and T. Nakajima, "Citron: A Context Information Acquisition Framework for Personal Devices," 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05), Hong Kong, China 2005.

[9] A. B. Davis, M. M. Moore and V. C. Storey, "Context-aware communication for severely disabled users," In Proceedings of the conference on universal usability (CUU'03), Vancouver, British Columbia, Canada, November 10-11, 2003.

[10] B.E. Saghir and N. Crespi, "A generic layer model for context-aware communication adaptation," In: IEEE Wireless Communications and Networking Conference, WCNC 2008, vol. 9(1), pp. 3027–3032 (2008).

[11] A. Ranganathan and Hui Lei, "Context-aware communication," Computer , vol.36, no.4, pp. 90- 92, April 2003.

[12] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, "ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications," IEEE Pervasive Computing, vol.4, no.2, 51-59, April 2005.

[13] N. Eagle and A. (Sandy) Pentland, "Reality mining: sensing complex social systems," Personal Ubiquitous Comput, vol.10, no.4, pp 255-268, March 2006.

[14] E. Horvitz, P. Koch, R. Sarin, J. Apacible, and M. Subramani, "Bayesphone: Precomputation of Context-Sensitive Policies for Inquiry and Action in Mobile Devices," User Modeling, Edinburgh, Scotland, July 2005.

[15] E. R. Pedersen, "Calls.calm: enabling caller and callee to collaborate," extended abstracts on Human factors in computing systems (CHI'01), pp 235-236, New York, NY, USA, 2001.

[16] Y. Nakanishi, K. Takahashi, T. Tsuji, and K. Hakozaki, "iCAMS: A mobile communication tool using location and schedule information," IEEE pervasive computing mobile and ubiquitous systems, vol.3, no.1, pp 82-88, 2004.

[17] S. Herborn, H. Petander and M. Ott, "Predictive Context Aware Mobility Handling," International Conference on Telecommunications, June 2008.

[18] K. Farrahi, D. Gatica-Perez, "Probabilistic Mining of Socio-Geographic Routines From Mobile Phone Data," IEEE Journal of Selected Topics in Signal Processing, vol.4, no.4, pp 746-755, August 2010.

[19] D. R. de Almeida, C. S. Baptista, E.R. da Silva, C. E. C. Campelo, H. F. de Figueirêdo, Y. A. Lacerda, "A Context-aware System Based on Service-Oriented Architecture," Proceedings of 20th International Conference on Advanced Information Networking and Applications (AINA'06), Vienna, Austria 2006.

[20] H.-L. Truong, L. Juszczyk, S. Bashir, A. Manzoor, S. Dustdar, "Vimoware - a Toolkit for Mobile Web Services and Collaborative Computing," Special session on Software Architecture for Pervasive Systems, 34th EUROMICRO Conference on Software Engineering and Advanced Applications, Parma, Italy, 3-5 September 2008.

[21] H. L. Truong, S. Dustdar, S. Corlosquet, C. Dorn, G. Giuliani, S. Peray, A. Polleres, S. Reiff-Marganiec, D. Schall, M. Tilly, "inContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms," In International Symposium on Applications and the Internet (SAINT' 08), Turku, Finland 2008.

[22] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems," PhdThesis, University of Maryland, Baltimore County, December 2004.

[23] P. L Cheng, K. Kakousis, N. Paspallis, G. A Papadopoulos, J. Lorenzo, and E. Soladana, "White Paper: MUSIC & Android," IST-MUSIC Deliverable (D13.11), 2009.

[24] RESTlet: RESTful web framework for Java, http://www.restlet.org/

[25] B. Chihani, E. Bertin, F. Jeanne, N. Crespi, "Context-Aware Systems: A Case Study," Proceedings of International Conference on Digital Information and Communication Technology and its Applications (DICTAP'11), Dijon, France 2011.