International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(1): 15-24
The Society of Digital Information and Wireless Communications, 2011 (ISSN 2220-9085)

HEP: context-aware communication system

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

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

ABSTRACT

Users may sometimes be overloaded with work so that they become temporally unable to handle incoming communications. Context aware systems are a promising approach to facilitate daily-life activities, especially providing assistance for communication management. After having surveyed the different challenges to build context-aware systems (e.g. modeling, reasoning about context), we introduce here HEP, a system that recommends communication services to the caller based on the callee's context. The recommendation is in form of advertising the workload of callee. HEP's main context source is the usage history of the different communication services as well as the users' calendars. It has been prototyped and tested at Orange Labs.

KEYWORDS

Ubiquitous Computing, Context-Awareness, Recommendation Systems, Communication.

1 INTRODUCTION

Nowadays, the user environment is fully embedded with smart devices integrating intelligence for processing various kinds of data. This ubiquity makes the interaction and management of all various devices that a user may hold a tough task. Context-aware systems are an emerging solution to alleviate such tasks; they will be in charge of supervising the way users interact with the ubiquitous environment for automating repetitive actions or adapting the execution of a service. For example, a context-aware system may use the location of user to enhance the quality of the result returned by Google search engine (e.g. if user looks for restaurant, the system will return only nearby ones). Such systems are fed with information, called 'context', about the user situation (e.g. location, activity, mood, etc), his social network (e.g. relationship with other people: family, friends) and surrounding environment (e.g. level of noise). For a network, context can be QoS (Quality of Service) parameters like RTT (Round-trip-Time). For a device, context can be its capabilities, display features or battery level.

Due to the computation complexity of managing all possible pieces of context information, the management and use of context are two decoupled functionalities. Thus, context-aware systems rely on CMS (Context Management System) to get the only needed subset of this context information.

From the conceptual viewpoint, CMSs are mainly based on the Producer-Consumer design pattern where context sources (e.g., sensors) play the role of

Producers, and context-aware applications play the role of Consumers. From the implementation viewpoint, CMSs can be classified into centralized or distributed architecture. In the centralized architecture, a central point often named broker is introduced between the producers and consumers. All context requests are handled by the broker, which forwards it to the right component. Producers and consumers are then decoupled, while in the distributed architecture, the different components have to know each other (e.g., by regularly sending multicast or broadcast messages for announcing themselves).

Many challenges face the field of context-awareness ranging from the collection of contextual information with the use of sensors (e.g. calendar, light, battery charge, etc.), to the modeling of context that can be anything (e.g. GPS location or a street address, time, etc.) and reasoning about it to produce an adaptive behavior (e.g. automated call transferring, the proposal of a meeting session, etc.). Starting from a real use case, our work shows how the challenges can be addressed when only relevant subset of the user's context is chosen.

In this paper, we first present the current researches in context-aware systems, by analyzing the ongoing works and the key issues for designing context-aware systems. We then introduce the major application fields of context-awareness. We finally present a case study on HEP, a centralized context-aware recommendation system designed, prototyped and experimented at Orange Labs, that takes as context information the usage of communication services.

2 DESIGNING CONTEXT-AWARE SYSTEMS

2.1 General overview

From the functional viewpoint, context-aware systems can be represented as a layered framework [1] (figure 1) composed from bottom to up by: sensors, raw data retrieval, preprocessing, storage/management, and application layer. The Context Management System (CMS) is responsible of retrieving raw data from sensors, abstracting and combining the sensed data into high level context, and then of making it available for context-aware applications.

The first layer (Sensors) is a collection of sensors responsible of retrieving raw data from the user environment (e.g. user device, social network, or used access network).

The second layer (Raw data retrieval) makes use of specific API or protocols to request data from the sensor layer. These queries should be implemented in a generic way, making possible to replace sensors (e.g. replacing a RFID system by a GPS one).

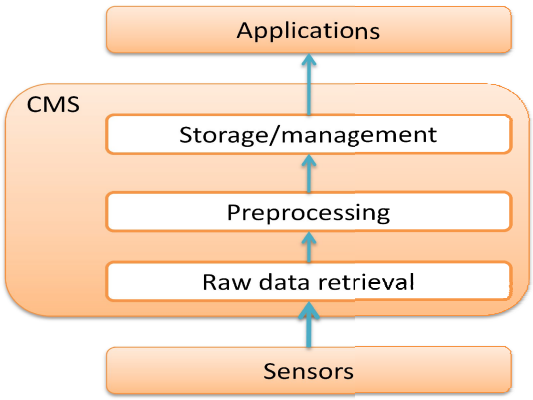

Figure 1. Layered framework for context-aware systems

The third layer (Preprocessing) is responsible for reasoning and interpreting contextual information. It

transforms the information returned by the underlying layer to a higher abstraction level (e.g. it transforms a GPS position to a position like at home or at work). Not only sensed or deduced data have to be modeled, but also meta-data describing them (e.g. accuracy and recall, or life cycle information).

The fourth layer (Storage and Management) organizes the gathered data and make them available to 3rd parties applications in a synchronous or asynchronous way. In the first mode, the 3rd party applications use remote method calls for polling the server for changes. In the second mode, they subscribe to specific events of interest, and are notified when the event occurs (for example by a call back).

The fifth layer (Application) is where the reactions to context changes are implemented (e.g. displaying text in a higher color contrast if illumination turns bad). We will now highlight some key aspects of CMS.

2.2 Context modeling

Many works tried different approaches for context modeling [2]. Most of these works distinguish between context information modeling (e.g. UML) and implementation technologies (e.g. XML).

Key-Value data structure [3] (where the key is the context, and the value is the corresponding sensed information) is the simplest representation. However, it lacks of richness, and do not support interoperability and the representation of relation among context information. XML-based languages are interesting candidates for modeling context because they rely on a widely used standard that provides the possibility to hierarchically represent context and to abstract it from low to high level. Much work (e.g. ContextML [4]) was done to propose a generic XML-based language for both context modeling and implementation.

MDA (Model Driven Architecture) is another interesting approach for the development of CMS [5]. It enables to create high level UML models or strictly speaking MOF (Meta-Object Facility) compliant models of the system, then based on these models, the implementation stubs are automatically generated alleviating tremendously the work of developers. UML-based modeling language (e.g. ContextUML) offers the full power of object orientation (encapsulation, reusability, inheritance) and also design flexibility by separating the modeling of context and context awareness from the service components.

RDF [6] (Resource Description Framework) schema has been extensively used for context modeling. Some vocabularies have been standardized on top of RDF to define context profiles like CC/PP [7] (Composite Capability/Preference Profile) and UAProf [8] (User Agent Profile). They have been combined with other modeling languages like FOAF[1] (Friend of a Friend) for modeling Person Organization, Group, Document and Project; vCard[2] for modeling addresses and personal data, Basic Geo for modeling geo-spatial context, vCal[3] for modeling events, ResumeRDF[4] for modeling skills and expertise of team members, and the Time ontology for modeling temporal context. However, the RDF language suffers from some limitations for the reasoning aspects, and

[1] http://www.foaf-project.org/
[2] http://www.w3.org/Submission/vcard-rdf/
[3] http://www.imc.org/pdi/
[4] http://rdfs.org/resume-rdf/

current work is more and more relying on ontology.

Modeling context with ontology allows a semantic description of context and sharing a common understanding of the context structure among users, devices and services. It also allows formal analysis of domain knowledge, i.e. reasoning using first order logic. OWL[5] (Web Ontology Language) is an ontology language based on a RDF schema. It enables to define rich vocabulary and to describe complex ontologies. OWL ontologies have been extensively used for context modeling, for instance in CoBrA [9] ontology (COBRA-ONT).

XML, RDF and OWL-based approaches are open and interoperable. Particularly, RDF and OWL offers the reuse of the common vocabularies while for XML there is no standard way for exchanging vocabularies. Associating RDF schema with OWL ontology can increase the expressiveness of the context description by drawing the relationship between a low level context information (e.g. the user is present in room 528) and high level one (e.g. the user is attending a meeting). As for UML, it is not directly compatible with XML/RDF/OWL, but it presents the advantage of being seamlessly integrated with MDE (Model Driven Engineering). This point is especially interesting when the whole CMS is designed using the MDE software approach.

It is interesting to store historical context data because it can be used to establish trends and predict future context values. Relational databases are usually used for context storage as plenty of available libraries allow the serialization of XML, RDF or OWL data.

[5] http://www.w3.org/TR/owl-features/

2.2 Quality of context

Context information can be retrieved from different kind of sensors having different level of reliability. Also, noise or failure of sensors can introduce imperfection on the sensed context. Other types of imperfections are ambiguity, imprecision, error. The notion of QoC (Quality of Context) aims to measure the imperfection of sensed information. A good example of QoC modeling is [10]. The authors propose an extendable UML-based model for context quality. They define three context levels (sensor, abstracted context and situation). For each level a set of quality parameters is defined. For example, Precision at the sensor level to indicate the maximum deviation of a measurement from the correct value, and Confidence at the situation level to assess the truthiness of the corresponding situation.

2.4 Context reasoning

Inferring new knowledge (e.g. transportation mean) from raw sensed data (e.g. GPS position) is important for context-awareness and adaptation to the user's context changes. But before being able to infer any new knowledge, some processing has to be done. Context processing can be divided into aggregation and interpretation. The former refers to the composition of raw context information either to gather all context data concerning a specific entity or to build higher-level context information. The later refers to the abstraction of context data into human readable information.

The inference can be done with help of sophisticated reasoning techniques that relies mainly on context representation. For example, SPARQL-based semantic reasoning techniques can easily be done

if the context representation technique is based on OWL. Ontology learning techniques can be used to derive new facts given a knowledge base of specific facts and an ontology describing concepts and relations among them. Machine learning techniques (e.g. Bayesian networks, fuzzy logic) can be used to construct higher level context attributes from sensed context. Combing both reasoning techniques can be interesting as demonstrated in [11]. Expert Systems (e.g. JESS, CLIPS) or Rule inference engines can also be used in context reasoning. Such reasoning systems inherit from forward-chaining inference the power of inferring knowledge (i.e. logical consequences) from sensed data (i.e. facts) and from backward-chaining inference the power of recognizing relevant context (i.e. facts). Knowledge might also be deduced using the Jena framework that provides ontology inference facility, and Jess (Java Expert System Shell) to implement forward-chaining inference. Jess is used when it is not possible to reason about context information with only ontology axioms, as described in [12].

Reasoning techniques are not widely supported. Also, OWL representations are hardly manageable (implementation /integration) and reasoning on XML data or UML class diagrams is not very developed. Reasoning with logical expressions like in Expert Systems allows a rich description of situations, actions and knowledge derivation due to the use of logical connectives (*and*, *or* and *not*), implications, universal and existential quantifiers.

3 CONTEXT-AWARE APPLICATIONS

Context-awareness aims to provide applications the ability to understand user needs, by analyzing his contextual information, and to adapt their behavior accordingly to meet user expectations. The adaptation may be performed in the service selection [13], in task execution [14], in security (e.g. applying an access control given a situation), in communication (e.g. selecting communication protocols [15]), or in content adaptation (e.g. adapting content resulting from a request [16]).

Context-aware applications can be: Location-Based Services (LBS), Context-Aware Communication (CAC), or Context-Aware Recommendation Systems (CARS).

LBS use location as the main contextual information for adapting accordingly the returned service. But, they may also combine outdoor localization (e.g. GPS) with social information (e.g. list of friends) and augmented reality technologies (e.g. Layar[6]) in order to help people for locating friends and places (e.g. restaurant) like in Nulaz [17], Foursquare[7], Gowalla[8], Loopt[9]. Also, enabling a location-based messaging like in Socialight[10], InfoRadar [18], Heresay [19]. Such services allow users to leave, in some GPS localization, an electronic message that can be read by anyone who is located nearby.

Context-aware communication (CAC) applications apply knowledge of people's context to reduce communication barriers [20]. For example, handling an incoming call based on the user availability (e.g. redirect calls to voice box if user is in meeting).

[6] http://www.layar.com/
[7] http://foursquare.com
[8] http://gowalla.com/
[9] http://www.loopt.com/
[10] http://socialight.com/

Context-aware homes [21] are another promising study field. Houses will be fully embedded with sensors and intelligent devices in order to support healthier everyday life of users: phones will ring only in the room where the callee is located to avoid disturbing everyone in house; lights and sound will be automatically adjusted based on the user who is present in the room; family member will be able to communicate as if they were in the front of each other, even if they are in different rooms; assistance of older people will be enhanced and their health conditions will be continuously assessed.

Context-Aware Recommendation Systems (CARS) aim to recommend a service or a product to a user based on his context (e.g. recommending movies [22]). For the recommendation to be relevant, CARS need to collect and to process a great amount of data (about products rating, users preferences, historical data, etc.) to predict the most relevant product or service to a user. In this paper, we have applied CARS concepts to communication services, by processing data retrieved from the Microsoft communication suite.

4 CASE STUDY: HEP

4.1 Usage scenario

Nowadays we live a big enhancement in communication tools, and there are many ways that can be used by people to communicate (e.g. email, phone). 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 (e.g. location, activity, and workload). The published information will help his contacts to evaluate the impact of the interruption caused by initiating a communication with him.

In this aim, we have developed HEP a context-aware system for recommending communication means for enterprise employees. The system publishes real-time information describing their status, emotions, activities and workload. The published information are results of processing diverse input streams concerning the usage of communication services (phone, IM, e-mail, calendar). The nature of the inputted information as part of the user context, how it is retrieved and how it will be processed make CMS the suitable management system, and context-aware system the suitable kind of application.

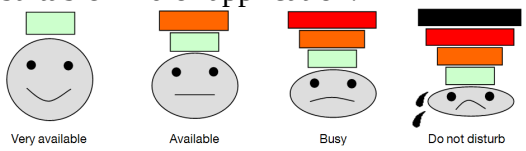

Figure 2. HEP statuses

Figure 2 presents the different statuses of a user: "Very available" corresponds to the state where user is highly available for receiving communication requests (e.g. phone call, IM request); "Available" corresponds to the state where user can receive call requests; "Busy" corresponds to the state where user can weakly respond to a call request; "Do not disturb" corresponds to the state where user cannot respond and will potential refuse incoming communication requests.

A status corresponds to the level of availability of a user on a given communication service (e.g. agenda, email, instant messaging, phone). Such information is used by the caller to

decide if he can interrupt the callee, and if it is better to use a communication service (e.g. email) than another service (e.g. phone) in order to reach the callee. For instance, let us suppose that Alice wants to call Bob for an urgent matter. Bob is at this moment in a conference call, but he is still reading his emails and answering them. With HEP, Alice will see that Bob is busy on the phone, but available by email. She decides thus to send his an email instead of calling him, although her demand is urgent.

4.1 Service design

Our system (figure 3) is developed in .Net and is based on OCS 2007 (Office Communication Server). The different elements composing the architecture are: a PC client, an Outlook plug-in, and a broker.

The PC client implements the two first functions of a CMS. It is responsible for Retrieving raw data from virtual sensors placed on Microsoft communication suite (Email, Calendar, Instant Messaging, fixed telephony). It is also responsible for computing user status for communication mean, and interacting with the broker.

The Outlook plug-in provides the user interface. It enables the user to set his preferences (e.g. the status that should correspond to a given load level), and above all it enables the user to see the statuses of each of his Outlook contacts.

The CMS storage and management functions are implemented in the broker that offers a directory service. PC clients subscribe and publish their status. And Outlook plug-ins requests the status of other users. An administration interface is available to set global rules for status computation.

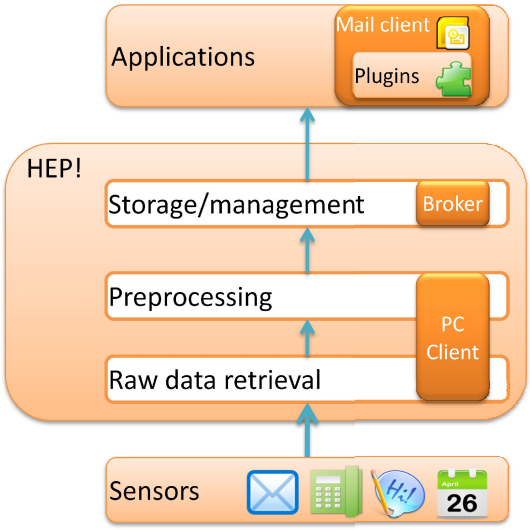

Figure 3. HEP architecture

Computation of user status is based on information about the history of usage of communication services (e.g. email) and desktop applications (e.g. Word, Excel, PowerPoint). The frequency of computation and freshness of status depends on the related communication service, but can be fixed by users when they specify their preferences.

4.3 Context modeling

We gather the different contextual information (both sensed and deduced ones) in an UML data model as illustrated in Figure 4. The BaseObject class is the root class in the model that is common to any type of context information. The InteractiveCom class gathers the shared attributes of interactive communication tools, while information specific to a communication tool is gathered in a specific class (e.g. Instant Messaging, email, phone). A specific class is dedicated to calendar information.

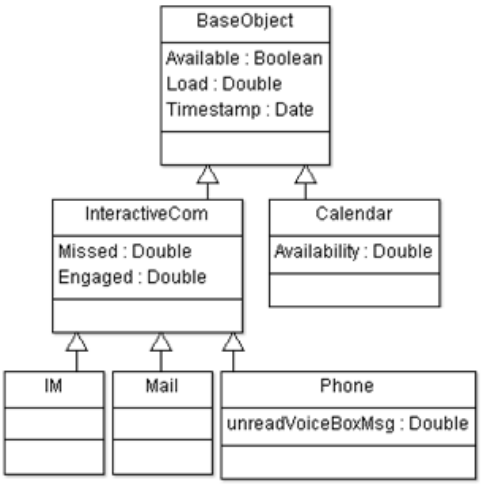

Figure 4. HEP Data Model

The description of the different attributes is as follow:

Available: is a sensed data that represents whether or not the user is available at a given instant in a communication tool (e.g. for calendar it can be interpreted as the user is currently not in a meeting).

Load: is a deduced data that represents work load of user corresponding to a communication tool (e.g. for calendar, load is the ration of the total amount of meeting time to work time), it correspond directly to the user status (figure 2).

Timestamp: represents for how long the sensed information remain valid, it depends on communication tool (e.g. 5mn for Mail, 15mn for Calendar);

Missed: represents the ratio of missed communication requests (e.g. missed phone calls, IM requests or unread mails) to the received.

Engaged: ratio of engaged communication to the received ones;

Availability: ratio between free time and total amount of meeting.

UnreadVoiceBoxMsg: ratio of unread message from the user's voicemail to the stored ones.

4.4 Context reasoning

The sensed information are used to compute the work load of a user on a given communication tool in order to determine the user status and whether or not he can accept incoming requests on this communication tool (figure 5). We defined rules for calculating the work load level for each communication mean (IM, mail, phone, calendar).

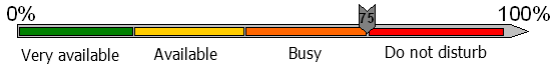

Figure 5. User status based on his work load level

In the case of calendar, if the user is currently in a meeting then we set his calendar work load to 100%. Then, the more the meeting start time gets closer, the more the calendar work load goes higher (e.g. 5m before a meeting, work load reaches 75% and user status become 'busy'). If the user is not in a meeting then calendar work load is the ratio of meeting duration in the rest of the day to the remaining work time.

Follow is an example of the calculation of the calendar work load of a user at different time of day. We consider that a work day start at 8:00 and finish at 18:00, and the user have a first meeting from 9:00 to 11:00 (2h duration), then a second one from 15:00 to 18:00 (3h duration). Thus, at 8:00 work load is (2+3)/10 = 50%, from 9:00 to 11:00 work load is 100% (user in meeting), at 12:00 work load is 3/6 = 50%, at 14:00 work load is 3/4 = 75%, and between 15:00 and 18:00 work load is 100%.We add a layer of abstraction by introducing the user global status that reflects the global workload. It is computed by combining the status related to the different communication means, with predefined weightings that can be modified by the end-user.

4.5 Future work

HEP has been deployed on the workstations of our coworkers at Orange Labs (Caen, France), and we received very positive feedbacks. The integration with the everyday working tools (e.g. Outlook) was especially approved. From the implementation viewpoint, several lessons can be derived.

The current reasoning technique is built with a set of IF-THEN clauses implemented in a C# class. We believe it is enough for a proof of concept solution, and we plan to use more sophisticated techniques like those provided by rule engines. For the context modeling language, we used an UML data model to take benefit of encapsulation and inheritance. The current modeling approach do not include metadata especially QoC parameters, we plan to include them in our future works. We found that these parameters are as important as sensed data themselves especially for managing very common situations where software crashes.

In our current solution, context processing including reasoning are performed at the client side. Such solution makes the deployment of new reasoning techniques for new included context data (e.g. about the usage of other office tools) more complicated. To overcome this issue we are planning to transfer a part of the preprocessing layer (figure 1), namely the reasoning part, to the broker side.

5 CONCLUSION

In this paper we surveyed the previously conducted works in the field of context-aware systems from different viewpoints like system design, context modeling and reasoning. After having highlighted the challenges to face when building context-aware systems and the major application fields for such systems, we proposed a context-aware system for recommending communication means. Our system helps users to choose the appropriate communication mean for contacting a person based on the context of the later. The aim behind the developed prototype is to build a context-aware system with the existing approaches in sensing, modeling and reasoning, to use it as a solution for a real problem, and experiment it with users in a real environment.

Besides the enhancements introduced in the previous section (lessons learned), we plan to expand our system with the ability to transform, in a transparent way, the format and the delivery time of a message based on the user's context. A message sent as an SMS at time *t* could be received, for instance at time *t + t'* as an email, given that the user is unreachable at *t* but may be reached at *t + t'* by e-mail only because he/she is still on the phone. We believe that this could lead to a new and seamlessly way to use our daily communication means.

6 REFERENCES

1. Loke, S.: Context-Aware Pervasive Systems: Architectures for New Breed of Applications. Auerbach Publications (2006)
2. Truong, H.L., Dustdar, S.: A survey on context-aware web service systems. International Journal of Web Information Systems, Vol. 5 Iss: 1, pp.5 - 31, (2009)
3. Yamabe, T., Takagi, A., Nakajima, T.: Citron: A Context Information Acquisition Framework for Personal Devices. Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05), (2005)
4. Venezia, C., Lamorte, L.: Pervasive ICT Social Aware Services enablers. 14th Int.

Conf. Intelligence in Next Generation Networks (ICIN). Berlin, Germany (2010)
5. Sheng, Q.Z., Benatallah, B.: ContextUML: a UML-based modeling Language for model-driven development of context-aware Web services. Proceedings of the International Conference on Mobile Business, ICMB'05 (2005)
6. RDF Working Group: Resource Description Framework (RDF). World Wide Web (W3). URL{http://www.w3.org/RDF/} (2004)
7. Kiss, C.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0. W3C Draft (2007)
8. UAProf Wireless Application Protocol WAP-248-UAPROF-20011020-a. WAP Forum (2001)
9. Chen, H., Finin, T., Joshi, A.: An Ontology for Context-Aware Pervasive Computing Environments. Journal of The Knowledge Engineering Review, Volume 18 Issue 3. Cambridge University Press New York, NY, USA (2003).
10. McKeever, S. Ye, J., Coyle, L., Dobson, S.: A Context Quality Model to Support Transparent Reasoning with Uncertain Context. 1st International Workshop on Quality of Context (QuaCon). Stuttgart, Germany (2009)
11. van Sinderen, M.J., van Halteren, A.T., Wegdam, M., Meeuwissen, H.B., Eertink, E.H.: Supporting context-aware mobile applications: an infrastructure approach. IEEE Communications Magazine (2006)
12. Ramparany, F., Benazzouz, Y., Martin, M.B.: Agenda Driven Home Automation - Towards High Level Context Aware Systems. Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC'09). Brisbane, QLD (2009)
13. Truong H.-L., Juszczyk L., Bashir S., Manzoor A., Dustdar S.: 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 (2008)
14. Truong, H.L., Dustdar, S., Corlosquet, S., Dorn, C., Giuliani, G., Peray, S., Polleres, A., Reiff-Marganiec, S., Schall, D., Tilly, M.: inContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms. In International Symposium on Applications and the Internet, SAINT'08. Turku, FINLAND (2008)
15. Herborn, S., Petander, H., Ott, M.: Predictive Context Aware Mobility Handling. International Conference on Telecommunications. St. Petersburg, Russie (2008)
16. Zimmermann, A.: Context Management and Personalization. PhD Thesis. University of Aachen (2007)
17. Pannevis, M.: I'm bored! Where is Everybody? Location Based Systems for Mobile Phones. MCs Thesis, University of Amsterdam (2007)
18. Rantanen, M., Oulasvirta, A., Blom, J., Tiitta, S., Mäntylä, M.: InfoRadar: group and public messaging in the mobile context. In Proceedings of the third Nordic conference on Human-computer interaction, NordiCHI '04, pp. 131-140 (2004)
19. Paciga, M., Lutfiyya, H.: Herecast: an open infrastructure for locationbased services using WiFi. IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'05). Montreal, Quebec (2005)
20. Schilit, B., Hilbert, D.M., Trevor, J.: Context-aware Communication. IEEE Wireless Communications (2002)
21. Meyer, S., Rakotonirainy, A.: A Survey of Research on Context-Aware Homes. Proceedings of Conference on Research and Practice in Information Technology, Vol. 21. (2003)
22. Bogers, T.: Movie Recommendation using Random Walks over the Contextual Graph. 2nd Workshop on Context-Aware Recommender Systems (CARS-10). Barcelona, Spain (2010)