Context-Aware Middleware Architecture for Smart Home Environment

Hamed Vahdat-Nejad, Kamran Zamanifar and Nasser Nematbakhsh

Computer Engineering Department, Faculty of Engineering,
University of Isfahan, Iran
vahdatnejad@gmail.com, zamanifar@eng.ui.ac.ir, nemat@eng.ui.ac.ir

Abstract

To realize smart home vision in an equipped domain, several types of context-aware applications should be deployed. Because of complex tasks of context gathering and processing, designing context-aware applications requires middleware support. Designing a context-aware middleware is a challenging issue because of specific characteristics of context and devices such as dynamic nature of context and resource limitation of devices. Furthermore, the middleware should provide a cooperative system in which application developers could easily exchange derived contextual information. To address these issues, in this paper architectural design of a smart-home context-aware middleware is proposed, which supports cooperation among application developers. The middleware is implemented as a Java package and validated through a case study.

Keywords- *Pervasive computing, Context-aware middleware, Context-aware application*

1. Introduction

Pervasive computing was initially introduced as the third era of computing [1], in which existing applications adapt their behavior to the context of the environment. Although numerous scenarios for designing context-aware applications have been proposed [2], designing stand-alone applications suffers from following deficiencies:

- Sensing the environment as well as gathering and processing context are complicated tasks, which make designing of context-aware applications much difficult and time-consuming.
- Most of the hosting devices (e.g. mobile phones, PDAs, tablets, etc.) have computational, storage and memory limitations and therefore, are incapable of efficiently running massive applications.

Context-aware middleware could address the above defects; however, designing such a middleware envisages new challenges, which have not been existed in the traditional middleware of distributed systems. Highly dynamic nature of context and limitations of hosting devices are among characteristics of the pervasive computing environment, which makes it impossible to customize and deploy similar middleware from distributed systems.

Previously, several studies have been conducted on context-aware middleware under the title of context management; however none of them fully addresses the characteristics of the pervasive computing paradigm. In this paper, architecture of a domain-based context management system is proposed. The architecture addresses known characteristics of the pervasive computing paradigm and provides a framework for facilitating cooperation among application developers. Context registration and retrieval are designed as services, which

should be invoked by context providers and consumers, respectively. The middleware is implemented as a minimal software package that should be imported by application developers. We evaluate and validate the package by investigating a case study.

Rest of the paper is organized as follows: after this introduction, Section 2 formally defines associated concepts and properly states the investigated problem through introducing a case study. Afterward, the state of the art is briefly presented. Section 3 describes the proposed architecture of the domain-based context-aware middleware. Subsequently, Section 4 presents investigation results of the middleware behavior in the described case study. Finally Section 5 discusses conclusion remarks and states future direction of this research.

2. Problem Definition and State of the Art

In this section, at first we provide utilized definitions and then describe a smart home scenario, which motivates this research. According to an initial comprehensive definition, "context is any information that can be used to characterize the situation of an entity. An entity is a place, person or object that is considered relevant to the interaction between a user and an application" [3, 4]. Context is of two types: low-level context involving time, location, identity, and activity, which are the most utilized contextual information, and high-level context, which involves any other information that can be derived from the low-level types [4]. Context source is referred to any device that can produce contextual data. Mobile phone, sensor, thermometer, pc, server, network, web, etc. are typical context sources.

A context-aware application always utilizes contextual information of the involved entities to adapt its behavior and provide context-aware services. Context-aware middleware aims to support designing context-aware applications via providing context management services. In general, context aware middleware lies on top of the operating system of mobile phones, PDAs, personal computers, servers, laptops, and other computational devices of the environment to and provides context management services [5].

2.1. Case Study

The following case study, which is originated from a smart home domain, helps in understanding requirements of a domain-based context management system.

Bob's house consists of a bedroom, living room, kitchen, and bathroom. It is augmented with sensors and smart devices. Bob is looking for various types of context-aware applications to help him in doing daily tasks and managing devices available in the domain. In continue three of these applications are described:

- ***Light adjustment application****: This application is responsible for adjusting light level of the rooms by turning lamps on or off. It uses a predefined context set consisting of time, presence of people in any of the kitchen, rooms and bathroom, Bob's location, Bob's activity, and then according to their value performs a suitable action. For example, during day as well as when Bob is outside the building or is asleep, the application turns the lights off. When Bob is watching TV, it adjusts the living room light moderately and turns off the other lamps. Finally, when Bob is in travel or vacation, the application randomly turns a few lights on to represent a lively state of the house (for security reason).*

- ***Mobile incoming call notification:*** *This application utilizes time (e.g. morning, midnight, weekend), Bob's location (e.g. bedroom, bathroom, outside at office), activity (e.g. sleep, watching TV, at work), and mobile contact list (to distinguish caller), and accordingly*

decides to redirect a receiving call to the voice-box, makes a vibration, or a slow, regular, or high alarm.

- ***Smart gate application:*** *Smart-gate application is responsible for handling arriving people to the house and is triggered by ringing the house bell. It utilizes a pre-installed small screen and video camera outside of the gate for taking picture of the person or establishing a video call between them and Bob. It uses time (e.g. morning, midnight, weekend), location (bathroom, outside), and activity (vacation, driving) of Bob and performs an activity such as opening the door (for relatives, familiar people, or a recognized parcel deliverer), makes a call between Bob's mobile phone and the person, etc.*

Afterward, Bob makes contract with three software developers who are hereafter referred as A, B, and C. They are assigned to implement Light adjustment, Mobile incoming call notification, and smart gate application, respectively.

2.2. Discussion

The above scenario prompts designing a system infrastructure for supporting development of context-aware applications. This kind of software, which is generally referred as "context-aware middleware", should provide a cooperative environment in a way that application developers could publish their derived context and also retrieve context published by others.

Previously, several researchers investigate context-aware middleware; however, they mostly concentrate on mobility of entities [6, 7, 8, 9], modeling [10-16], reasoning [17-21], context dissemination [22-26], quality of context assessment [27-31] and privacy protection [32-33]. The aim of this research is to investigate context-aware middleware toward providing a cooperative system for developing context-aware applications.

3. Smart-home Context Management

In this section architecture of the proposed domain-based context-aware system is described. The system is minimal in the meaning that it focuses on context gathering, aggregation and distribution. For the sake of simplicity, we eliminate autonomous functional requirements of the middleware such as quality of context assessment, modeling and privacy protection, which require an independent research investigation. However, the architecture is extensible and these tasks could be inserted to the system later.

Since mobile devices have limitations in memory, availability and processing capabilities, a dedicated server is exploited as host of the massive part of the middleware.

The server maintains a table, which keeps an overall view of the context types generated in the domain as well as access points of their providers. The table has three columns: name of the entity whose situation is characterized by the context, context type, and access point of the provider.

An interface of the middleware is provided on all the computational devices available in the domain; therefore, they perform massive operations by invoking associated remote methods from the server. Figure 1 shows a schema of the middleware architecture. In the following a detailed description of the components and functionalities is provided:

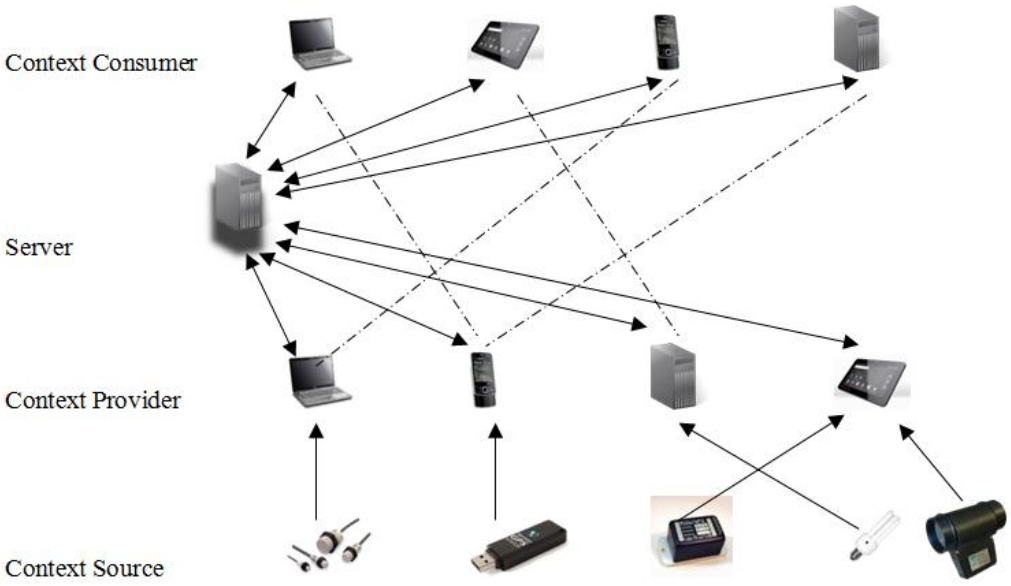


Figure 1. Overall Architecture of the Smart-home System

- *Context provider*: one type of context provider is a program module that contacts a context source, gathers raw data, processes and models it. Another kind of context provider involves programs that infer complicated context types from context provided by several context sources and providers. Programs that exploit sophisticated context mining techniques for deriving complicated context types are also categorized as context provider. Context providers might reside on any computational device or server. They can contact each other by their access point address.

- *Registration service*: registration service, which is implemented on the server, is invoked by context providers to register their contextual capacities. The service gets three parameters, which involves type of the provided context, name of the entity that the context is associated to, and access point of the context provider's machine. Therefore it updates the table by inserting a new tuple.

- *Context retrieval service*: any context-aware application in the initialization phase utilizes the context retrieval service provided by the server to obtain address of the devices providing its contextual needs. This service gets entity name and context type and returns access point of the machine hosting the corresponding context provider.

3.1. Implementation

We make use of Java as a portable language for implementing the middleware. The middleware is provided as a Java package, which is installed on the server. The server contains a table, which is responsible for keeping a general view of the context types available in the domain. The schema of the table as well as an example of its values is shown in figure 2. The following methods are the main modules of the middleware. They are implemented on the server and invoked remotely by application developers via Java RMI.

- Register (entity name, context type, access point): This method is used by context providers to register their contextual capacities. It gets context type, entity name, and access point of the context provider and inserts the corresponding tuple to the table.

- Retrieve (entity name, context type): This method is exploited by application developers to find their required context. It gets context type and name of the associated entity and returns access point of device(s) providing this context element.

Entity-Name	Context-Type	Context Provider Access-Point

Figure 2. Schema of the Table

4. Evaluation

In this subsection, we investigate behavior of the proposed context-aware middleware through studying the presented case study scenario. Before starting, the use case investigation, it should be noted that any computational device has a clock, which indicates "time" context; therefore, context-aware applications acquire "time" from their host machine. Figure 3 illustrates the use case behavior of the system via presenting interactions of the components. In continue a detailed description of the events sequence is described:

- After analyzing the situation, software developer A decides to use the server as the host of Light adjustment application. Since this application needs presence context of any people in the kitchen, rooms, and bathroom, the developer decides to implement a module for providing this context type. The module contacts with noise and motion sensors installed in the rooms, kitchen, and bathroom, and gathers their value. Afterward, through simple reasoning, it infers presence context and registers it to the server by invoking "register" method.

- Mobile incoming call notification application resides on Bob's mobile phone. It requires Bob's location context. Software developer B realizes that the context is not provided in the system, so she decides to develop a program module for acquiring it. For this purpose, she uses a two level scheme. In the first level, the program uses longitude/latitude data provided by mobile phone GPS to estimate general location of Bob, such as home, office, university, shopping center, etc. if Bob is currently at home, the second level uses presence information to infer exact location of Bob in terms of living room, bedroom, kitchen or bathroom. Besides, by invoking "retrieve" method, software developer B realizes that presence information is already existed in the system and gets access point of the providing program, which resides on the server. Finally, by invoking register method, location of Bob is registered to the server.

- Smart gate application requires time, location and activity context. By invoking "retrieve" method, software developer C finds out location is available and gets access point of the providing program. However, he realizes that activity is unavailable, so he decides to implement a program for inferring it. For this purpose, he uses many of context types provided in the domain such as time, location, velocity (provided by GPS of Bob's mobile phone), noise level, presence, and home devices situation (TV is on, oven is off, etc). Most of these context types are previously provided and registered by other applications. After inferring activity, it is registered to the server, and the developer starts implementing smart gate application.

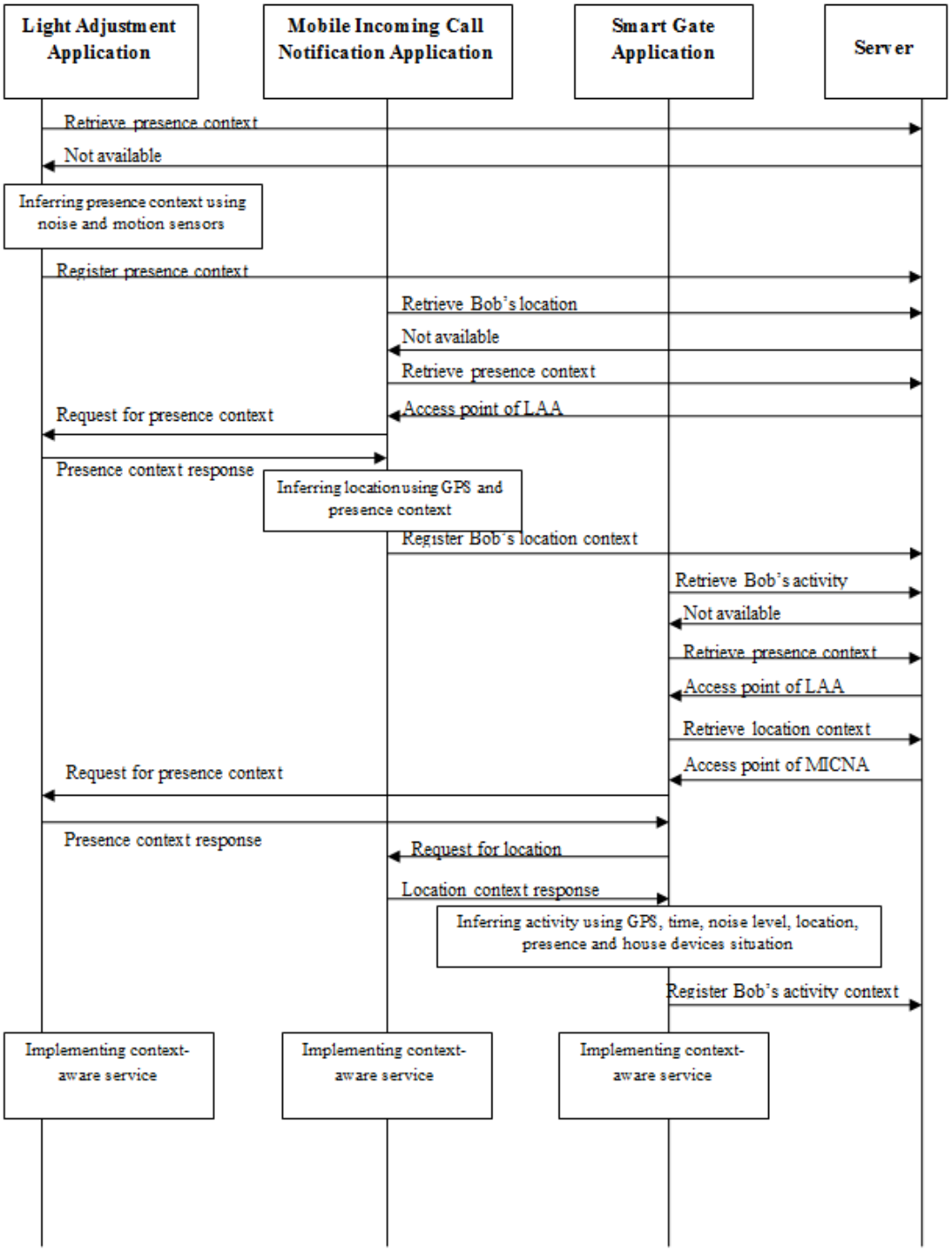


Figure 3. Case Study Behavior of the System

- By invoking "retrieve" method software developer A acquires all of its required context types; hence he starts to develop light adjustment application.
- The same situation is occurred for Mobile incoming call notification application.

4.1. Discussion

The presented scenario indicates a typical pervasive computing environment in which diverse context-aware applications require context types generated by each other. In continue we discuss the behavior of the system and advantages yielded comparing to the stand-alone design approach.

Although light adjustment application requires presence, location and activity context, it only infers presence and then shares it with others applications by registering it to the middleware layer. A similar situation is occurred for mobile incoming call notification application. Since, the application resides on Bob's mobile phone, which is incapable of running a massive program, it retrieves presence context and by using it, provides and registers location context to the middleware layer. Finally, by retrieving complicated activity context from the middleware, a much smaller program is achieved. The resulted mobile incoming call notification application is much more suitable than a stand-alone application for being executed on a limited machine, because it eliminates massive parts of inferring presence and activity context.

Finally, software developer C finds every context required for developing smart gate application available, except activity. He retrieves presence and location context from the middleware layer and infers activity context. At last, he shares the activity context with the other applications by registering it to the middleware layer.

The investigated case study consists of only three applications; however, most of the sophisticated context types required by these applications are retrieved from the middleware layer. Since the number of context types available in an environment is confined and limited, as the number of context-aware applications increases, application developers find almost their entire required context types registered to the middleware. As a result, the proposed middleware yields to a cooperative system, which facilitates sharing of context provider modules throughout the system.

5. Conclusion

In this paper a context-aware middleware system has been presented to facilitate sharing contextual information via diverse entities available in a smart home environment. The middleware has provided easy to use mechanisms for context publishing and retrieving to make it possible for software developers to provide new sophisticated context types by utilizing previous available context elements, which are offered by other entities. As a result, it tends to lead to a pervasive computing environment with various types of sophisticated context-aware applications, which are actually simple and dependent programs.

More general pervasive computing environments consists of a group of domains such as mobile, ad hoc, and user personal and also physically limited to a geographical zone domains such as home, office, hospital, university, city, etc. Designing a context-aware middleware for a multiple-domain environment envisages new challenges such as need for a distributed architecture, aggregating context generated in different domains, and mobility of entities between domains. In the next step of the current research, we are going to address these challenges by designing a multiple-domain context-aware middleware.

Acknowledgements

This work was partially supported by Iranian Telecommunication Research Center associated to Ministry of Information and Communications Technology [grant number T/500/10180]. We would like to thank them for giving us the opportunity for doing this work.

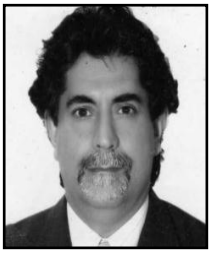
References

[1] M. Weiser and J. Brown, "The Coming Age of Calm Technology", in Beyond calculation: the next fifty years, P. Denning and R. Metcalfe, Eds., New York, Copernicus, **(1997)**, pp. 75 – 85.
[2] "Context aware computing group", MIT media lab, http://www.media.mit.edu/context/.
[3] A. K. Dey, "Understanding and using context", Personal and Ubiquitous Computing, vol. 5, no. 1, **(2001)** February, pp. 4-7.
[4] A. K. Dey, *et al.*, "Towards a better understanding of context and context-awareness", in Lecture Notes In Computer Science, Germany, **(1999)**.
[5] H. Vahdat-Nejad, K. Zamanifar and N. Nematbakhsh, "Towards a Better Understanding of Context Aware Middleware: Survey and State of the Art", To be published, **(2013)**.
[6] B. Guo, L. Sun and D. Zhang, "The Architecture Design of a Cross-Domain Context Management System", in 8th IEEE International Conference on Pervasive Computing and Communications Workshop, Mannheim, **(2010)**.
[7] A. Cadenas, *et al.*, "Context Management in Mobile Environments: a Semantic approach", in Proceedings of the 1st Workshop on Context, Information, and ontologies, Greece, **(2009)**.
[8] C. Hesselman, *et al.*, "Bridging Context Management Systems for Different Types of Pervasive Computing Environments", in Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, Brussels, **(2008)**.
[9] P. Pawar, *et al.*, "Bridging Context Management Systems in the Ad Hoc and Mobile Environments", in IEEE Symposium on Computers and Communications, Sousse, Tunisia, **(2009)**.
[10] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich and D. Chakraborty, "Intelligent Agents Meet the Semantic Web in Smart Spaces", IEEE Internet computing, vol. 8, no. 6, **(2004)** November 15, pp. 69 - 79.
[11] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi and D. Zhang, "Semantic Space: An infrastructure for smart spaces", IEEE Pervasive Computing, vol. 3, no. 3, **(2004)** July, pp. 32 - 39.
[12] J. Euzenat, J. Pierson and F. Ramparany, "Dynamic context management for pervasive applications", The Knowledge Engineering Review , vol. 23, no. 1, **(2008)** March, pp. 21-49.
[13] Z. Jaroucheh, X. Liu and S. Smith, "CANDEL: Product line based dynamic context management for pervasive applications", in International conference on complex, intelligent and software intensive systems, Krakow, Poland, **(2010)**.
[14] I. Satoh, "A location model for smart environments", Pervasive and Mobile Computing, vol. 3, no. 2, **(2007)**, pp. 158-179.
[15] L. Seremeti, C. Goumopoulos and A. Kameas, "Ontology-based modeling of dynamic ubiquitous computing applications as evolving activity spheres", Pervasive and Mobile Computing, vol. 5, no. 5, **(2009)** October, pp. 574–591.
[16] S. L. Amundsen and F. Eliassen, "A resource and context model for mobile middleware", Personal and Ubiquitous Computing, vol. 12, no. 2, **(2008)** January, pp. 143-153.
[17] Y. Oh, J. Han and W. Woo, "A context management architecture for large-scale smart environments", IEEE Communications Magazine, vol. 48, no. 3, **(2010)** March, pp. 118 - 126.
[18] B. Guo, D. Zhang and M. Imai, "Toward a cooperative programming framework for context-aware applications", Personal and Ubiquitous Computing, vol. 15, no. 3, **(2011)** March, pp. 221-233.
[19] A. Kofod-Petersen and M. Mikalsen, "Context: Representation and Reasoning - Representing and Reasoning about Context in a Mobile Environment", REVUE D'INTELLIGENCE ARTIFICIELLE, Trondheim, **(2005)**.
[20] C. Anagnostopoulos and S. Hadjiefthymiades, "Enhancing Situation-Aware Systems through Imprecise Reasoning", IEEE Transactions on Mobile Computing, vol. 7, no. 10, **(2008)** October, pp. 1153 - 1168.
[21] Y. Jiang, Y. Tang, J. Wang and S. Tang, "Representation and reasoning of context-dependant knowledge in distributed fuzzy ontologies", Expert Systems with Applications, vol. 37, no. 8, **(2010)**, pp. 6052-6060.
[22] N. Roy, T. Gu and S. K. Das, "Supporting pervasive computing applications with active context fusion and semantic context delivery", Pervasive and Mobile Computing, vol. 6, no. 1, **(2010)** February, pp. 21-42.
[23] H. Vahdat-Nejad, K. Zamanifar and N. Nematbakhsh, "A New Approach to Context Distribution in Large-Scale Environments", in Proceedings of the International Conference on Trends in Information Technology and Applications, Ajman, UAE, **(2010)**.
[24] T. -S. Chen, G. -J. Yu and H. -J. Chen, "A framework of mobile context management for supporting context-aware environments in mobile ad hoc networks", in Proceedings of the international conference on Wireless communications and mobile computing, Honolulu, Hawaii, **(2007)**.
[25] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis and N. Liampoti, "Optimising context data dissemination and storage in distributed pervasive computing systems", Pervasive and Mobile Computing, vol. 6, no. 2, **(2010)**, pp. 218 - 238.

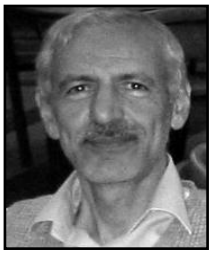
[26] H. Vahdat-Nejad, K. Zamanifar and N. Nematbakhsh, "A New Community-Based Context Distribution Approach for Large-Scale Pervasive Systems", International journal of ad hoc and ubiquitous computing, **(2012)** August.
[27] P. Hu, J. Indulska and R. Robinson, "An autonomic context management system for pervasive computing", in Annual IEEE International Conference on Pervasive Computing and Communications , Hong Kong, **(2008)**.
[28] S. Becker, *et al.*, "Reference Model for the Quality of Context Information", University of Stuttgart, Stuttgart, **(2010)**.
[29] K. Sheikh, M. Wegdam and M. V. Sinderen, "Middleware Support for Quality of Context in Pervasive Context-Aware Systems", in Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, New York, **(2007)**.
[30] P. Pawar, A. Van Halteren and K. Sheikh, "Enabling Context-Aware Computing for the Nomadic Mobile User: A Service Oriented and Quality Driven Approach", in Proceedings of IEEE Wireless Communications and Networking Conference, Kowloon, **(2007)**.
[31] A. Manzoor, H. -L. Truong and S. Dustdar, "Quality Aware Context Information Aggregation System for Pervasive Environments", in International conference on advanced information networking and applications , Bradford, United Kingdom, **(2009)**.
[32] A. Toninelli, A. Corradi and R. Mon, "A quality of context-aware approach to access control in pervasive environments", in Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Berlin, **(2009)**.
[33] J. I. Hong and J. A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing", in Proceedings of the 2nd international conference on Mobile systems, applications, and services, Boston, **(2004)**.

Authors

Hamed Vahdat-Nejad is currently a PhD candidate at the computer engineering department of the University of Isfahan. In 2011, he was a research visitor at the Sapienza University of Rome in the MidLab laboratory. He received his master degree from Ferdowsi University of Mashhad in 2007, and his bachelor's degree from Sharif University of Technology in 2004. His research spans pervasive and context aware computing. He published about 15 refereed journal and conference papers from 2006 to 2011. He has experiences in chairing and organizing special sessions and workshops. He is founder and chairman of the "International Workshop on Context Aware Middleware for Ubiquitous Computing Environments" Series. He is Editorial board member of the "Journal of Software Engineering", published by Academic Journals and also "Lecture Notes in Information Science and Technology". He has served as TPC for IWCMC, ISIEA, ICCIT-WCS, CAM.



Kamran Zamanifar received his Bs and MSc from Tehran University in electronic engineering. He got his PhD in parallel and distributed systems form Leeds University in UK. He is an associate professor at computer engineering department, the University of Isfahan. His research interests are in parallel processing, distributed systems and pervasive computing.



Nasser Nematbakhsh received his Bs degree from University of Isfahan in 1973, Ms degree from Worcester Polytechnic Institute, USA in 1978, and PhD in Computer engineering from University of Bradford, UK in 1989. He is currently associate professor in the computer engineering department of the University of Isfahan. His main research interests include Pervasive computing, Performance Evaluation, and Software Engineering Methods, Tools & Techniques.