Steps Towards Pervasive Software: Does Software Engineering Need Reengineering?

Dana Al Kukhun, Florence Sedes

IRIT, Paul Sabatier University, 118 Route de Narbonne, 31062 Toulouse, France.

Abstract. Nowadays, the definition of service is demanding machines to turn into human beings. In order to work efficiently, machines need to analyze current situations, figure out problems or challenges and find out solutions. Aiming to achieve context-aware or adaptive pervasive systems, the software and hardware should accomplish an intelligent, automatic and proactive adaptation process that responds to current contexts. System performance will be guaranteed only if we add new terms to its behavior, such as: self-adaptation, self-organization, self-configuring, self-healing, self-optimizing and self-protecting. These challenging automated processes can produce proactive behavior if software engineers use the environment context as a solution instead of thinking about it as an obstacle.

Keywords. Pervasive systems, interactive systems, ubiquitous computing, software engineering, adaptation, context-awareness.

1 Introduction

Due to the revolution of Information Technology, a new computing era is taking place. Many challenges need to be met, especially in a mobile and dynamic environment where users are interacting with different devices and constructing ad hoc networks, while systems are supposed to provide them with proactive value-added services.

Pervasive or Ubiquitous Computing was first introduced by Weiser as his vision for the future of computing in the 21st century; where computing elements will disappear from the user's consciousness while functioning homogeneously in the background of his environment [13]. In pervasive computing, users communicate and compute with each others whenever, wherever and however [9]. Pervasive computing allows coupling the physical world to the information world and provides a wealth of services and applications that allow users, data, machines, applications and physical spaces to interact seamlessly with one another [1].

Pervasive computing merges physical and computational infrastructures in an integrated environment, where different computer devices and sensors are gathered to provide new functionalities, offer specialized services and boost productivity [15].

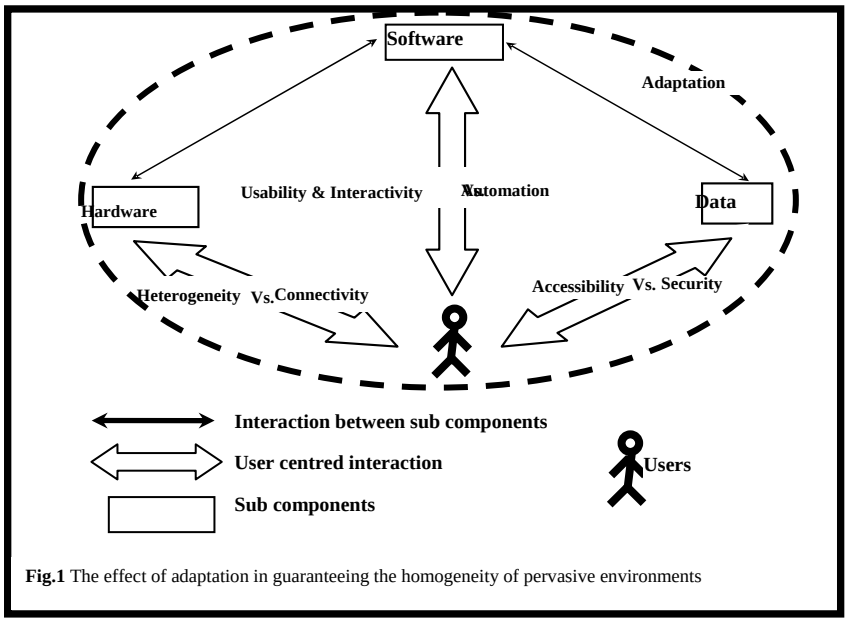

Fig.1 The effect of adaptation in guaranteeing the homogeneity of pervasive environments

While analyzing pervasive computing and studying its progression, it was found that for hardware and computing elements to disappear, software needs to disappear and the spatial and temporal relationships between people and objects (human-machine interaction) has to be well defined in the early design phase in order to cope with the dynamicity of the ubiquitous computing environments [17].

In this section, we have presented different definitions of pervasive computing. Next, in section 2, we'll present our vision that consists of framing these systems within defined boundaries of interactive environments; we analyze the components of these environments, the different features and the challenges that face the interactivity of these components from a user-centered perspective. In section 3, we'll present the problem of integrating the changing context information with the content information. In section 4, we present the enabling technologies that help the system to adapt with the heterogeneity of the software components. Then in section 5, we'll present some system requirements and challenges that the software should ensure. Finally, we present our proposition that changes the adaptively logic and uses the environment as a stimulating factor for adaptation instead of considering it as a burden.

2 Pervasive Computing Environments

Pervasive information systems are interactive user-centered systems that aim to facilitate user interaction within unfamiliar environments. Facilitating this interaction is highly recommended and comes, in our perspective, through three

principal components that should be interacting homogeneously with each others and with the user. These components are: data, software and hardware, see Fig 1.

Pervasive environments should adapt with the dynamic, changing and distributed computing systems that extend the boundaries of physical spaces, the building infrastructures and even the devices contained within these environments.

Pervasive computing environments should be aware of the context [2] and should be able to capture situational information in order to integrate them with users and devices. The interaction between these components should be as transparent as possible and this transparency can be applied using different metrics along with run-time, automatic adaptation for both content and context, starting from the early stage of requirements analysis and design, till late testing and execution of the system.

In a previous article [3], we have presented the sub components of pervasive environments and the different challenges that take place during their interaction. We have defined the challenges that encounter the software engineer in defining the relationship between the user and data, software and hardware.

In the user -- data interaction axe, the user wants to access the system easily and to obtain as much information as he needs. On the other hand, the system should be safe and secure in order to protect the information it contains and should only allow authenticated users to access the system. Therefore, there is a challenge between assuring the security and accessibility at the same time. We have highlighted the importance of applying multi layered adaptation in order to balance between the user requirements and the system security constraints.

Similar challenges are present in the user -- hardware relationship, where the hardware engineers want to provide users with small-sized, light-weighted, small screened devices that have high storage and processing capacities. At the other side of the system, users need to interact easily with the system without facing human-machine interaction difficulties that deprive him of smooth and efficient usage.

Finally, the user -- software relationship is also a confusing one; the user demands for flexible, usable and interactive systems that enable him to control the system behavior and as a result obtain a system that meets his changing needs. However, software engineers tend to provide users with automated services that react towards predicted situations and this way, the system will be guarantying its behavior and its reliability without worrying about user interaction and control.

In this article, we'll discuss the importance of the software component and how it should be developed in a way that facilitates the user interaction, accomplish the service provider objective and guarantee the user satisfaction by using the environment as an encouraging factor not a burden or a dynamic obstacle.

3 Pervasive Software Engineering: Between Context and Content

As the computer disappears, capturing the context of use is becoming of high importance. As the user moves around un-trusted environments (e.g. ad-hoc networks), the system should model the user in order to provide him with personalized information that adapts to his needs, interests and characteristics. In

the mean while, the system should capture the environmental context in order to provide services that turn the surrounding environments into intelligent spaces.

Capturing context is a challenging mission especially when we need to take under consideration different heterogeneous data sources and data formats. The data should be integrated and normalized with the aim of facilitating their processing and their interpretation so that they will be finally used to manage the context.

Context management was introduced as an effective tool for creating, integrating and administrating context aware applications [2]. This tool considers the definition of some relevant context parameters, the link between these parameters and the source of information they're taken from and how they're used to accomplish the targeted adaptive behavior.

Content can be connected to context using metadata; this process enriches the content and facilitates its interpretation. XML is used to handle the heterogeneity of hardware devices, software components and connection channels. The integration process needs to validate new interaction protocols between service providers and ad hoc connected users.

As pervasive systems use variant Commercial of the Shelf COTS hardware products, different connection channels interact with different system infrastructures and platforms. Thus, a crucial need is demanded for achieving interoperability.

4 Software Enabling Technologies

Software engineering is a vital axe in which programmers try their best to coordinate between the different parts of pervasive systems. As hardware devices are diverse, their configuration and operation programs are different and in this case different software components have to cooperate and achieve a common task.

Next, we present some useful software enabling technologies, standards and reference architectures that help facilitating the development of pervasive applications and assure that these services provide users with the content they desire and their context awareness. This way, services will become highly proactive.

4.1 Software Components

Constructing an intersection point of distributed systems, embedded systems and telecommunications, Pervasive computing poses many challenges to software engineering. Software development has become a very competitive field, especially in the case of pervasive systems where there is an increasing demand for highly productive and coast effective implementation techniques.

Software engineers have exploited the fact that pervasive systems could be distributed in different physical locations and thus introduced the usage of software components. A component is an independently deployable piece of software that resides on a hardware element and provides a service. Web services are components that reside on the server side, pervasive systems aims to apply the

concept of components in a peer-to-peer design architecture using a well defined protocol [18].

In order to go beyond the limits and turn computing applications into pervasive ones, the adaptation process should be performed automatically at run-time and should meet the application and context needs just as the user needs. Automatic adaptation can be done by employing TBA Type-Based Adaptation [18] which solves the incompatibility problem between the different interfaces by selecting a previously written adapter from an adapter repository and automatically determine the adapter to be combined in order to translate between the different components.

As pervasive environments allow users to interact using different "Commercial Of The Shelf" COTS products, the use of different software components will facilitate the integration of the different heterogeneous products. Meanwhile, such integration highlights the importance of providing a middleware technology that would help to provide a homogeneous environment instead of carrying out complex matching and adaptation processes.

4.2 Middle Ware

The adaptation process in pervasive systems has to deal with the context volatility and unpredictability so the adapter repository might be maintained at the communication end parties.

Adapting and matching different software components is only efficient if these components have similar architectures and if the adaptation functions are available at the server or the end parties. Consequently, software engineers thought about constructing middle ware technologies as a solution [5].

Pervasive computing connects many applications together so matching a lot of software components won't be a practical solution. On the other hand, transforming these components into a more generic and powerful middle ware would facilitate the integration of different components, their composition and finally assuring homogeneous communication.

Providing a uniform and adaptive middle ware technology will assure the interoperability between different services within ad hoc networks. Although standards, reference architectures and generic software technologies provide the basis for future ubiquitous software development, new kinds of micro architectures and software technologies and development methods are needed.

4.3 Programming Models

Establishing suitable programming models for pervasive spaces is essential in improving the productivity, enhancing the quality of pervasive systems, and creating an open platform for interoperability.

In order to increase the productivity, quality and interoperability, the implementation of a pervasive computing environment can follow one of two different programming models as follows:

1) ***Context - Driven Model*** where several contexts can be defined in advance using description logic. On runtime, the system will explicitly know the behavior that it should follow according to its current state; this model assures the interoperability,

extensibility and scalability of the system and is ideal for discovering contradictory system behaviors and to detect conflicts and adapt with multimodal environments.
2) ***Service - Oriented Model*** which is more expressive, proactive and procedural model that allows higher programming control levels and concentrates on the services that should be provided using several decision making techniques that memorize past actions along with the environment's history during a service lifecycle [8].

4.4 Software Factories

Software Factories [11] produce reusable assets that reduce the overall development time. MDA, Model-Driven Architecture, promotes the use of high abstraction level models which offer system developers with an intuitive way to describe the system.

Pervasive systems development is being a challenging mission because of the heterogeneity of used technologies which leaves the developer with a low abstraction level. MDA can be employed in the modeling process in order to cope with the technological heterogeneity problem and then Software factories can be used to raise the platform's abstraction level by providing an amount of code that is reasonably reduced.

MDA standard proposes to build generic PIMs, ***Platform Independent Models***, that would be then transformed into customizable PSMs, ***Platform Specific Models***; these models provide clear descriptions to applications and are finally translated into final source code.

A Perv-ML , ***Pervasive Modeling Language***, was proposed in [11] in order to provide a domain, UML-based, specific language that describes pervasive systems using high abstraction level constructs. Perv-ML promotes the separation of developer roles; where they can be either ***system analysts***; that look at the system details and construct high abstraction level constructs or ***system architects***; that consider the system general structure.

4.5 Software Design Patterns

In order to facilitate the design, implementation and development of pervasive systems, standard solutions to common problems in software design were generated and gathered together. Design patterns take a systematic approach, which focuses on the patterns of interaction instead of focusing on how individual components work,. Design patterns describe abstract systems of interaction between classes, objects, and communication flow.

Design patterns communicate insights into design problems, capturing the essence of recurring problems and their solutions in a compact form. Patterns capture design knowledge, such as guidelines and heuristics, in three ways. First, patterns offer low-level solutions to specific problems rather than providing high-level and abstract suggestions. Second, patterns are generative, helping designers create new solutions by showing many examples of actual designs. Third, patterns are linked to one another hierarchically (structured), helping designers address high-level problems as well as low-level ones. Patterns are intended to complement

guidelines and heuristics. Patterns are simply another tool for helping designers create high-quality solutions.

An experimental pervasive application was developed using design patterns and it was found that patterns helped novice designers to generate designs, helped experienced designers that are new to the ubiquitous computing domain to learn about the domain, helped designers to communicate ideas, and finally, helped designers to avoid potential design problems earlier in the design process [6].

4.6 XML & XACML – Generic Interoperable Standards

XML eXtensible Markup Language is a text based language for data classification that helped to accomplish better data integration and exchange. XML is rapidly becoming the standard for data representation and portability between systems.

The demanding need for security, privacy, authentication and access control has motivated the emergence of a new policy language XACML eXtensible Access Control Markup Language. In order to prevent unauthorized access attempts within open dynamic pervasive environments XACML had automated several managerial tasks and allowed interoperable interactions between several applications along with the reuse of access control policies [7].

XACML standard represents a policy language and an access control decision request/response language. Being an XML based standard language; XACML has succeeded to accomplish interoperable interaction with other applications.

XACML is a generic language that can be embedded and used in any environment. Its policies can be distributed in arbitrary locations. The power of XACML is due to its ability to support a wide variety of data types, functions and rules that enables combining the results of different policies [14].

5 Evolutionary Systems with High-Level Requirements

Pervasive computing has introduced new high level system requirements that should be taken in consideration in the design and implementation process.

Interoperability is highly demanded in all the levels of pervasive systems. Software components should be built independently of the context in which they are used in, this way they will be used in different computing environments and applications [5].

Heterogeneity is a challenge in pervasive environments is due to the context dynamicity and the user mobility, so the network connectivity is changing over time and mean while, the user – which has a dynamically evolving context and profile – is interacting with the system using different hardware devices. As a result the software will have to cope with different screen resolutions, user interaction methods, radio capabilities, memory capacities, power and processing capabilities.

Mobility is an important requirement that can take different forms. Actual mobility is the capability of an autonomous software agent to dynamically transfer its execution towards the nodes where the resources it needs to access are located. Exploiting this form of mobility will save network bandwidth and increase the

execution reliability and efficiency. This aspect can be deployed to help embedded software agents to follow mobile users wherever they go. Virtual agent mobility is the ability to be aware of the multiplicity of networked execution environments.

In this case, agents are aware of the distributed nature of the target, are explicitly locating and accessing resources using the best available connectivity option in the environment. Physical agent mobility is used when mobile and wireless computing devices access the resources by dynamically changing access points.

Survivability and security are demanding requirements that if provided will enforce the systems with powerful capacities. Survivability is the ability of a system to fulfill its mission on time despite the presence of attacks and failures. Survivable systems require a self-healing infrastructure with improved qualities, such as security, performance, reliability, availability and robustness [5].

An application may employ security mechanisms, such as passwords and encryption, yet may still be fragile by failing when a server or network link dies. On the other hand, an application must be able to survive malicious attacks and must, therefore, support security.

The literature has presented two kinds of survivability in the context of software security: survival by protection (SP) allows security mechanisms like access control and encryption to ensure survivability by protecting applications from malicious attacks. The survival by adaptation (SA) gives the application the ability to survive by adapting itself to the changing dynamic conditions.

The continuity is a very demanding feature in ubiquitous applications especially with the uncertainty and instability of user connectivity while he moves around. The application should have the possibility to pause the user session in the case of sudden disconnection and continue to work later without the loss of current state and the running history [4].

As we have shown, pervasive environments are highly dynamic environments so their software should provide application-aware adaptation that allows applications to determine the best adaptation behavior for the situation but preserves the ability of the system to monitor resources and enforce allocation decisions.

In the adaptation context, agility and evolvability become important requirements. Agility allows managing changes that are timely unpredictable but have predictable characteristics. Evolvability enables handling changes in the long-term during the life cycle of a system.

Self-organization is the ability of a system to spontaneously increase its organization without the control of the environment or external systems. Self-organizing systems not only regulate or adapt their behavior but also create their own organization. Self-organization applies the concepts of self-learning, expert systems, chaotic theory and fuzzy logic. Self-organization may be also applied in ad hoc networks in order to reach improved and efficient performance, minimize cost and increase reliability and survivability.

Usability, according to human-machine interaction engineers and to interactive system designers, is how easy an interface design is to be understood and used, how unambiguous interactive controls are and how clear its navigational scheme is [9]. This feature is the final objective of pervasive systems.

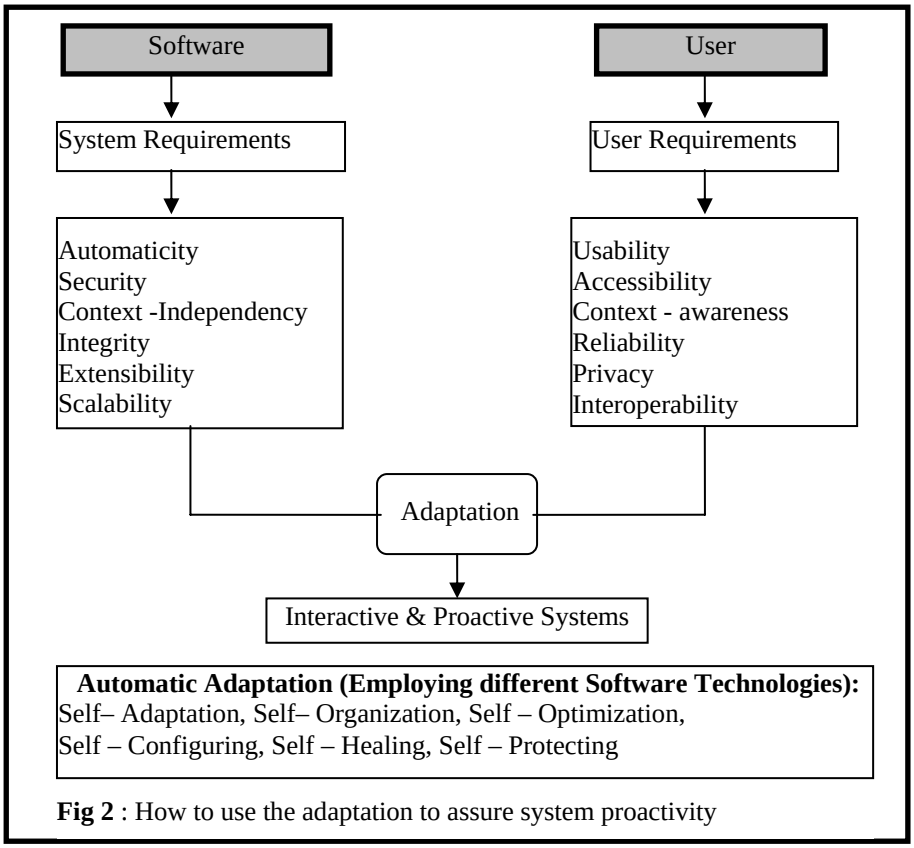

Fig 2 : How to use the adaptation to assure system proactivity

Next, we'll present a new vision that aims to ensure the aspired features within pervasive dynamic environments.

6 A Good Reflection: Using Proactivity to Adapt to The Context

Proactive services will introduce context aware applications that would analyze the capacities of the system and the surrounding dynamic environment then it will automatically anticipate some potential features that could be applied and thus provide the user with services that surpass his expectations. The system will employ the surrounding environment resources to assure better data integration, representation, visualization and processing. See fig 2.

Applications are still written with a particular display format in mind and do not have the ability to transform their layout from one set of display dimensions to another. As wireless capabilities become more prevalent on all devices, the use of a device's display may no longer be limited to the device it is physically attached to. A computer with an impoverished display can now consider broadcasting its display pixels across a wireless link to a computer with better screen quality. The

idea here is to highlight the importance of resource exploitation and that adaptation is demanded not only to provide the best services within the limitations of available resources but also to transform the disadvantages of the mobility into useful features where the surrounding environments will be exploited in order to assure proactive unexpected services.

Of course many security parallels should be resolved in order to allow using common resources in data visualization such as; data transfer, hardware accessibility, the presence of other users in the case of visualizing private data, etc. The use of location context goes beyond just knowing where you are, but can take into account 'who' is with you, further building contextual clues about the activity undertaken at that time [16].

As atomicity is highly demanded in the pervasive computing environment, users are being controlled by the application and deprived of obtaining personalized services or interfaces. As a result, some ideas have grown in the perspective of increasing the flexibility of user interaction with different applications.

This perspective was to increase the application availability by a new modeling architecture for user interfaces where the user interface layer is separated from the rest of application layers. This way application interfaces will support a high level of accessibility to all users and support a high degree of group individualization with more individualized user interaction without the need for specific software customization by software developers.

Separating the user interface execution platform and the application logic platform will provide maximum availability of applications to users, especially when these applications are used in small sized devices that don't need have a lot of space or importance for user interfaces. Accessibility to a wider range of users will be assured since the user interface will customized according to the users culture, language and it could even follow the user preferences or be compatible with his common user interface objects.

As a step to contextual deployment was to apply technical deployment of user interfaces in different devices, this step aims to achieve real internationalization and user individualization. The complete separation of the user interface from the rest of the application provides a ready made user interface solution for any application based on the Model Based Architecture.

Systems performance will be guaranteed if we add new terms to its behavior, such as: self-configuring, self-healing, self-optimizing and self protecting. In order to embed these options and employ them whenever needed, the process of sensing context should be application independent and the representation of sensed context should be as generic as possible. [9]

We propose to provide a hybrid Contextual-Service oriented model. This model will be analyzing the user context and as the context changes the system would carry out a recursive loop of a proactive adaptation process. As a result, the service will be accomplished in a way that responds to the user (context and needs) and would meet the service provider objectives. A dynamic requirements analysis would provide a successful adaptation process and might affect the structure of the "Software Development Life Cycle" SDLC.

Finally the automatic adaptation that we're looking for means that the service will be fully aware of context information that will be updated as the user moves and as his situation changes. Context information can contain the user location, current connectivity, the different software components (services) that the user is contacting, the hardware devices that he could be interacting with or that are available around him and finally, the most important component that the user needs to obtain which is the data content. To obtain this interactivity and proactivity, we should use the previously mentioned enabling software technologies that would ensure system ubiquity.

7 Conclusion

Being in a pervasive computing environment, everything becomes dynamic and heterogeneous; data, hardware, connectivity and software. On the other hand, the user stays static in his physical and logical abilities so the greatest mission is the one of the software that should coordinate between the different parties and adapt everything to the user profile, requirements and context. In this article, we present a new dimension of software adaptation techniques in which we propose using the context to help the system function proactively.

8 References

[1] A. Ranganathan et al, "Towards a pervasive computing benchmark", PerCom 2005 Workshops, Third IEEE International Conference on Pervasive Computing and Communications, 8-12 March 2005, pp.194-198.

[2] A. Zimmermann, A Lorenz, and M. Specht, "Applications of a Context-Management System", Modeling and Using Context: 5th International and Interdisciplinary Conference CONTEXT 2005, Paris, France, July 5-8, 2005. pp 556-569

[3] D. Al Kukhun and F. Sedes, "A Taxonomy for Evaluating Pervasive Computing Environments", Mulitmodal and Pervasive Services Workshop, IEEE Conference on Pervasive Services, 29 June 2006, Lyon, France.

[4] E. Chen, D. Zhang, Y. Shi, and G. Xu, "Seamless Mobile Service for Pervasive Multimedia", Advances in Multimedia Information Processing, PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, 30 Nov – 3 Dec, 2004. Proceedings, Part II.

[5] E. Niemela, J. Latvakoski, "Survey of Requirements and Solutions for Ubiquitous Software",ACM International Conference Proceeding Series; Vol. 83 a, Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, College Park, Maryland, 2004, pp 71 – 78.

[6] E. S. Chung et al., "Development and evaluation of emerging design patterns for ubiquitous computing", Symposium on Designing Interactive Systems, Proceedings of the 2004 conference on Designing interactive systems: processes, practices, methods, and techniques, Cambridge, MA, USA, 2004, pp 233 - 242 .

[7] F. Almenárez, A. Marín, C. Campo et C. García, "TrustAC: Trust-Based Access Control for Pervasive Devices", 2nd International Conference on Security in Pervasive Computing, Boppard, Germany, April 2005. pp 225-238.

[8] H. Yang, E. Jansen and S. Helal, “A Comparison of Two Programming Models for Pervasive Computing”, Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium, 23-27 Jan. 2006, pp134 – 137.
[9] I. Park, W. Kim and Y. Park, "A Ubiquitous Streaming Framework for Multimedia Broadcasting Service with QoS based mobility Support", Information Networking, LNCS 3090 in Springer-Verlag (SCI-E), June 2004, pp.65-74.
[10] J. Davis, A. Tierney, E. Chang, “A User-Adaptable User Interface Model to Support Ubiquitous User Access to EIS Style Applications”, COMPSAC, Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05), 2005, pp. 351 – 358.
[11] Munoz and V Pelechano, “Building a Software Factory for Pervasive Systems Development”, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005. pp 342-356
[12] L. Graham, “The principles of Interactive design”, Delmar Publishing, 1999.
[13] M. Weiser, “The computer for the 21st century”, ACM SIGMOBILE Mobile Computing and Communications Review, Volume 3, Issue 3, July 1999, pp: 3 - 11.
[14] OASIS, “A brief Introduction to XACML”, 14 mars 2003, http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html , last consulted on 25 nov. 2006.
[15] R. Campbel, J. Al-Muhtadi, P. Naldurg, G. Sampemane and M. D. Mickunas, “Towards Security and Privacy for Pervasive Computing”, Proceedings of International Symposium on Software Security, Tokyo, Japan, 2002.pp. 1-15.
[16] R. Want, T. Pering, “System challenges for ubiquitous & pervasive computing”, International Conference on Software Engineering, Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA, 2005, pp 9–14.
[17] R. Want, T. Pering, G. Borriello and K. Farkas, “Disappearing hardware”, Pervasive Computing, IEEE ,Vol 1, Issue 1, Jan.-March 2002. pp:36 – 47.
[18] T. Gschwind, M. Jazayeri and J. Oberleitner, “Pervasive Challenges for Software Components”, Radical Innovations of Software and Systems Engineering in the Future: 9th International Workshop, RISSEF 2002, Venice, Italy, October 7-11, 2002. pp 152-166.
[19] 19. Y. Duan and J. Canny, “Protecting User Data in UbiComp: Towards trustworthy environments”, Privacy Enhancing Technologies (PET 2004), May 2004, Toronto, Canada. Selected Papers. pp167-185.