Steps Towards Pervasive Software: Does Software Engineering Need Reengineering?

Dana Amin 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, perceive user needs and provide users with intelligent, automatic and proactive adaptation that responds to current contexts. System performance will be guaranteed only if we add new features 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 change the engineering logic and use the environment context as a solution instead of thinking about it as an obstacle.

Keywords. Pervasive 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, constructing ad hoc networks, while systems should 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 compute and communicate with each others whenever, wherever and however [9].

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

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 the challenges that face these systems. In section 3, we'll present the difficulties of integrating context within content information. In

section 4, we'll present some enabling technologies that help the system to adapt with the heterogeneity of its software components. In section 5, we'll present the challenges that meet software engineers while providing different system requirements. Finally, we highlight the importance of changing the adaptively logic and using the environment as a stimulating factor for adaptation.

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 to serve the user. These components are: data, software and hardware, see Fig 1.

Pervasive computing environments should be aware of the context and should be able to capture situational information. Transparent interaction between different components can be provided 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 the 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.

In the ***user -- data*** relatioship, the user full and easy access to information sources. 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 ensuring system security and accessibility. We highlight the importance of applying multi layered adaptation in order to balance between user requirements and system security constraints.

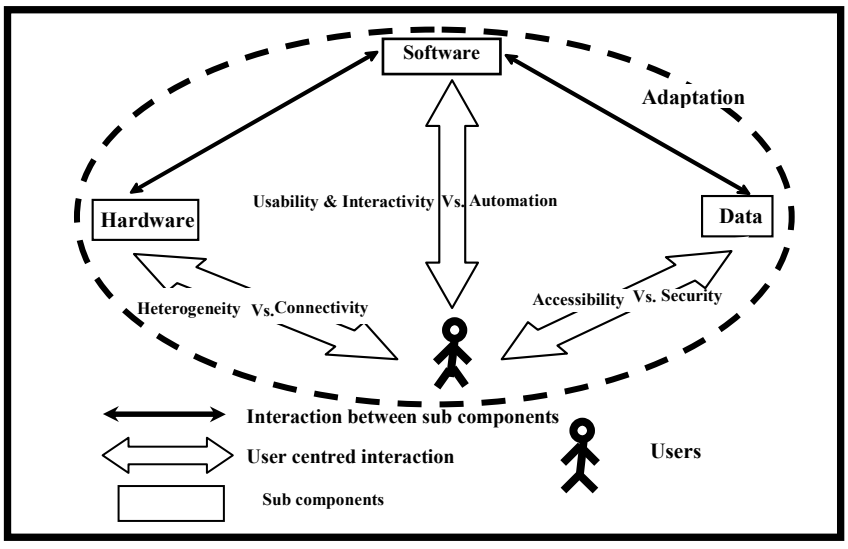

Figure 1. How can adaptation guarantee the homogeneity of pervasive environments

Similar challenges are present in the ***user -- hardware*** relationship, where 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, users need to interact easily with their machines 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.

3 Pervasive Software Engineering: Between Context and Content

As the computer disappears, capturing the context is becoming of high importance. When a user moves around un-trusted environments, the system should model the user in order to provide personalized services that adapts to his needs, interests and characteristics. Mean while, the system should capture environmental context to provide services that turn the surrounding environments into intelligent spaces.

Capturing context is a challenging mission especially from data sources. Data should be integrated and normalized with the aim of facilitating their processing and interpretation to manage context. Context management was introduced as a tool for creating, integrating and administrating context-aware applications [2].

Context can be connected to content 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 in order to achieve interoperable interaction.

3.1 Software Enabling Technologies

Software engineering helps programmers in coordinating between the different parts of pervasive systems. We present some software technologies, standards and reference architectures that facilitate the development of pervasive applications.

3.1.1 Software Components

Software engineers have exploited the fact that pervasive systems could be distributed in different physical locations and introduced software components. A ***component*** is an independently deployable piece of software that resides on a hardware element and provides a service. ***Type-Based Adaptation*** TBA is an automatic adaptation which solves the incompatibility problem between the different components and interfaces [18].

Deploying different software components will facilitate the integration of heterogeneous products. Meanwhile, such integration highlights the importance of middleware technologies that help to provide homogeneous and interoperable environments instead of carrying out complex matching and adaptation processes.

3.1.2 Middleware

The adaptation process in pervasive systems has to deal with its context volatility and unpredictability. Pervasive computing connects many applications together. Matching a lot of software components is not a practical solution but transforming them into generic and powerful middleware would facilitate the integration of these components and would ensure homogeneous communication [5]. A uniform and adaptive middleware technology would ensure interoperability between different services within ad hoc networks.

3.1.3 Programming Models

In order to increase the productivity and interoperability, the implementation of a pervasive application can follow one of two 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*** is an expressive, proactive and procedural model that allows higher programming control levels and employs decision making techniques to provide a service or compose a new service from already existing ones by consulting the history of the service lifecycle [8].

3.1.4 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 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 heterogeneity and then Software factories can be used to raise the platform's abstraction level by providing a reduced amount of code.

3.1.5 Software Design Patterns

Patterns facilitate systems design by providing solutions to common interaction problems between classes, objects, and communication flows. Patterns offer low-level solutions to specific problems rather than providing high-level and abstract suggestions. Patterns are generative; they help designers to create new solutions by showing actual designs examples. Finally, patterns help designers to solve problems of different structures wheather high-level problems or low-level ones. Patterns are intended to complement guidelines and heuristics [6].

3.1.6 XACML – A Secure Interoperable Standard

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, XACML provides automated managerial tasks, allows interoperable interactions between several applications with the reuse of access control policies [7].

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

3.2 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, this way they will be used in different computing environments and applications.

Heterogeneity is a challenge in pervasive environments; mobile users interact with the system using different hardware devices, the context and connectivity become dynamic. Meanwhile, the user also has a dynamically evolving profile. As a result the software should provide services that adapt with different screen resolutions, user-interaction methods, machine power and processing capacities.

Mobility is an important requirement. ***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[5].

Survivability and ***security*** provide systems with powerful capacities. Survivability is the ability of a system to fulfill its mission on time despite the presence of attacks and failures. Such functioning requires a self-healing infrastructure with improved qualities such as security, reliability, availability and robustness. 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. 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 [5].

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 losing information [4].

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.

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

4 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 would 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 ensure better data integration, representation, visualization and processing. See fig 2.

Adaptation should not only provide the best services within the limitations of available resources but should also transform the disadvantages of the mobility into useful features where the surrounding environments will be exploited to ensure proactive unexpected services.

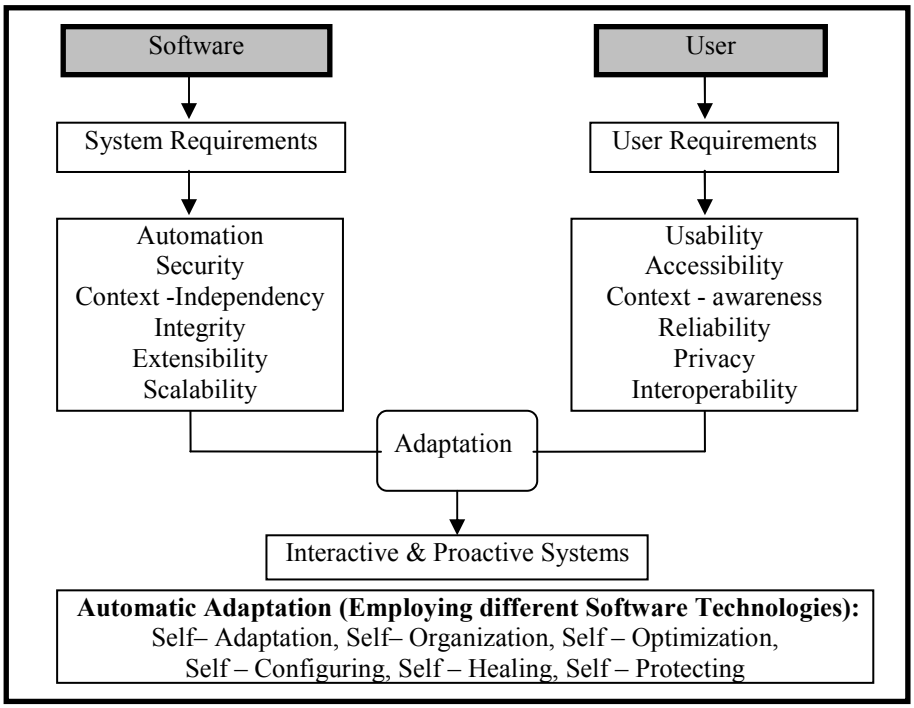

Figure 2. How to use the adaptation to assure system proactivity

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.

Atomicity is highly demanded in the pervasive computing environment, but users are being controlled by the application and deprived of obtaining personalized services or interfaces. Therefore, user interaction flexibility is needed. This perspective increases the application availability by a new modeling architecture for user interfaces where the user interface layer is separated from the rest of application. 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 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 ensured since the user interface will be 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.

We propose to re-engineer the modelling of pervasive systems by following 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 process that aims at searching for the proper proactive adaptation process that might reside already in the service repository or might be composed from different behaviours. 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 is the user who needs to obtain access to the system or just needs a service. To obtain an interactive and a proactive behaviour, we should use a combination of the previously mentioned enabling software technologies that would ensure system ubiquity and transparency.

7 Conclusion

In a pervasive computing environment, everything becomes dynamic and heterogeneous; data, hardware, connectivity and software. The software's mission is and to coordinate between the different system subcomponents in order to satisfy the changing needs, requirements and user 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] Al Kukhun D and Sedes F, "A Taxonomy for Evaluating Pervasive Computing Environments", MAPS 2006, IEEE Conference on Pervasive Services, 2006, pp 29-34.
[2] Almenárez F, Marín A, Campo C and García C, "TrustAC: Trust-Based Access Control for Pervasive Devices". In The 2nd International Conference on Security in Pervasive Computing, Germany, 2005, pp 225-238.
[3] Campbel R, Al-Muhtadi J, Naldurg P, Sampemane G and Mickunas MD, "Towards Security and Privacy for Pervasive Computing". Software Security, 2002 p 1-15.
[4] Chen E, Zhang D, Shi Y and Xu G, "Seamless Mobile Service for Pervasive Multimedia". In PCM'04, IEEE, 2004, p 194-198.
[5] Chung ES, Hong JI, Lin J, Prabaker MK, Landay, JA and Liu AL, "Development and evaluation of emerging design patterns for ubiquitous computing". In 2004 conference on Designing interactive systems, USA, p 233 - 242.
[6] Davis J, Tierney A and Chang E, "A User-Adaptable User Interface Model to Support Ubiquitous User Access to EIS Style Applications". COMPSAC'05, IEEE, p 351–358
[7] Duan Y and Canny J, "Protecting User Data in Ubiquitous Computing : Towards trustworthy environments". In PET 2004, Springer, p 167-185.
[8] Graham L, "The principles of Interactive design", Delmar Publishing, 1999.
[9] Gschwind T, Jazayeri M and Oberleitner J, "Pervasive Challenges for Software Components". In RISSE 2002, Springer, p 152-166.
[10] Munoz J, Pelechano V, "Building a Software Factory for Pervasive Systems Development". Advanced Information Systems Engineering, Springer 2005, p 342-356
[11] Niemela E and Latvakoski J, "Survey of Requirements and Solutions for Ubiquitous Software". 3rd International Conference Mobile and Ubiquitous Multimedia, pp 71–78.
[12] OASIS, "A brief Introduction to XACML", 2003, access on 4/2007, available at www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML
[13] Park I, Kim W and Park Y, "A Ubiquitous Streaming Framework for Multimedia Broadcasting Service with QoS based mobility Support". In Information Networking, Springer, 2004, p 65-74.
[14] Want R, Pering T, Borriello G and Farkas K, "Disappearing hardware". In Pervasive Computing, 2002, IEEE, p 36 – 47.
[15] Want R, Pering T, "System challenges for ubiquitous & pervasive computing", Software Engineering Conference, 2005, p 9–14.
[16] Weiser M, "The computer for the 21st century", ACM SIGMOBILE Mobile Computing and Communications Review, 1999, p 3-11.
[17] Yang H, Jansen E and Helal S. "A Comparison of Two Programming Models for Pervasive Computing". In SAINT 2006, IEEE, p134 – 137.
[18] Zimmermann A, Lorenz A and Specht M, "Applications of a Context-Management System", In Modelling and Using Context, Springer, 2005, p 556-569.