2009 33rd Annual IEEE International Computer Software and Applications Conference

A Perspective on Middleware-oriented Context-aware Pervasive Systems

Zakwan Jaroucheh, Xiaodong Liu, and Sally Smith
School of Computing
Edinburgh Napier University
{z.jaroucheh, x.liu, s.smith}@napier.ac.uk

Abstract

The evolving concepts of mobile computing, context-awareness, and ambient intelligence are increasingly influencing user's experience of services. Therefore, the goal of this paper is to provide an overview of recent developments and implementations of middleware-based pervasive systems, and to explore major challenges of implementing such systems. This paper also provides a comprehensive access to the literature of the emerging approaches and design strategies of middleware for providing users with personalized services taking into consideration their preferences and the overall operating context. Middleware systems were categorized according to their internal coordination model.

1. Introduction

With the great growth of mobile devices such as laptops, palmtops, and smart phones, it is hardly surprising that the mobile computing has attracted considerable attention in recent years. In an attempt to go beyond the traditional view of explicitly used computers and terminal devices, a new more general paradigm of user-centric mobility has been emerged. In this paradigm, the "Ubiquitous Computing" concept which has been introduced by Mark Weiser in 1991 [8], a smart and autonomous computing technology will be embedded in every device to [15] enhance the use of computers by making computers effectively available throughout the physical environment and, at the same time, making them invisible to the user. Mark Weiser [8] expressed this goal as achieving the most efficient technology and making computing as ordinary as electricity. Thus, instead of relying on specialized devices carried and maintained by the user such as mobile phone, the focus is now on provisioning services to the user [3].

Accordingly, the main objectives of this review are:

- To highlight the pillars of the pervasive computing as well as relevant technologies standards and research areas.

- Laying the groundwork for developing context-aware systems.

- To highlight different approaches for personalization of services and their interaction, and how these approaches can take advantage of current software engineering techniques.

The remainder of this paper is structured as follows. Section 2 discusses the current distributed systems. Section 3 describes the fundamental requirements of context-aware pervasive systems. Section 4 presents an abstract architecture of middleware-based context-aware systems. Section 5 presents the role of middleware. In Section 6 we outline context-aware service discovery, service composition and service adaptation. Finally, Section 7 concludes the paper with brief concluding remarks.

2. Today's Distributed Computing

The evolution of distributed computing was driven by the new pervasive networking. Network connectivity is embedded in most digital resources thanks to the widespread of the Internet, the availability of broadband and wireless networks, and the convergence of telecom and computer networks. Two application areas were emerged that exploit such network connectivity [10]. These are grid and ubiquitous computing which further drive the middleware evolutions and research.

2.1 Grid Computing

Grid computing is a form of distributed computing that depends on software to divide and allocate pieces of a program among several (sometimes thousands) networked, loosely coupled computers, therefore creating a "super and virtual computer" acting to perform very large tasks.

The Grid is a truly heterogeneous environment where the resources are geographically distributed and managed by a multitude of institutions and organizations. It offers a persistent, standard-based service infrastructure for the purpose of creating a distributed community that share resources such as storage space, software applications and data [10]. In this respect, the objective of the resource sharing, achieved by direct and coordinated access to resources, is to achieve collaboration problem solving.

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.142

2.2 Pervasive Computing

The vision of "disappearing technologies", pioneered by ubiquitous computing [8], adopts the idea of building smart environments where computing and communication facilities are "everywhere", being embedded in almost every surrounding object and can be seamlessly accessed "every time".

This vision is becoming reality [10] thanks to the great evolution of software and hardware technologies (e.g., wireless networks, mobile computing, and agents).

3. Requirements of Pervasive Context-aware Systems

Context-aware applications bring to light several architectural design challenges. In the following we present some of the new kind of requirements that have to be treated by the pervasive context-aware systems:

3.1 Coordination

In decentralized setting of pervasive environments, resources need to be able to coordinate, from the application to the network layer. Furthermore, coordination must be supported in a decentralized way to avoid reliance on a specific infrastructure whose accessibility cannot always be guaranteed in mobile and open network [10]. In other words, without a single controlling entity, the coordination of adaptations among individual components or subsystems becomes a major design consideration, with cross-cutting impacts on numerous functional and non-functional properties [16].

3.2 Interoperability

The vision of pervasive computing requires a very large range of devices and software components to interoperate seamlessly. Interoperability is required at all levels of ubiquitous computing [15]. On the application level, mobile client applications must discover and interoperate with application services available to them at their present location. However, these services will be developed upon a range of middleware types and advertised using different service discovery protocols unknown to the application developer [18]. Therefore, interoperability on the middleware level is also required.

3.3 Heterogeneity/Transparency

In ubiquitous world, we can identify two main types of heterogeneity:

- *Hardware heterogeneity*: Applications must be aware of two kinds of heterogeneity [11]. The first one relates to the computational environment offered by the context in which they are running (e.g. the bandwidth offered by the communication medium). The second one concerns the device itself: the same communication infrastructure can be accessed by a large variety of devices which are all homogeneous in terms of basic resources and functionalities, but different with respect to quantitative and qualitative characteristics (e.g., memory size, computational power, etc.).

- *Software heterogeneity*: heterogeneity of software is expressed by a diversity of software structures, component models, interface technologies and languages [15].

3.4 Mobility

The focus in mobile computing research should consider four types of mobility: device mobility, "personal mobility", service mobility, and code mobility. In general, mobility results in context change which may affect how clients consume the services and how the provider serves them [19].

3.5 Survivability & Security

In ubiquitous world, the behavior of components may be unpredictable because of changes in the environment (e.g. network connection failure, services failure to provide the expected functional and/or non-functional qualities, etc.). This is why we see an increasing interest towards self-healing solutions [7]. A key characteristic of a survivable system is its ability to deliver essential services even in the face of attack, failure or accident [15].

3.6 Adaptability

The need for adaptability in software is growing, driven in part by the emergence of pervasive and autonomic computing [14]. Software services must adapt to different kinds of terminals and networks. They also have to handle dynamically emerging and evolving contexts and user preferences. Due to mobility in ubiquitous environment, dynamically changing conditions make adaptability a big challenge [15].

3.7 Autonomic Behavior

Autonomic computing becomes critical to managing mobile devices at the wireless edge, and also in managing large-scale computing centers.

Traditional instructive systems with their passive, deterministic, context-free, and pre-programmed nature are suggested to be replaced by autonomous computing systems, which are active in nature and implement non-deterministic, context-dependent, and adaptive behaviors.

3.8 Scalability and Resource-discovery

Pervasive systems may include a large number (potentially tens of thousands) of distributed components or subsystems [16]. The mobility and availability of a potentially this infinite number of heterogeneous resources at the same time entail requirements such as scalability and resource-discovery [10].

3.9 Context-awareness

3.9.1. Defining Context-awareness

Context can be defined as [4] meta-information to characterize the specific situation of an entity, to describe a group of conceptual entities, and to partition a knowledge base into manageable sets or as a logical construct to facilitate reasoning services.

As stressed by the ubiquitous vision, distributed systems need not only to adapt to the change in the available resources, but also to the users' preferences and profiles over time and the physical environment. This ability is generally referred to as context-awareness [10].

Obviously, context-awareness is central to ubiquitous computing that aims at delivering applications to end-users in an opportunistic way, with the best quality possible. Therefore, development of context-aware systems requires: context management as well as context-based adaptation [10].

Furthermore, in ubiquitous environment, we can define context-awareness as [12]: the capability of a context-aware system or middleware to provide anytime access to heterogeneous, distributed, and unanticipated context information in global scale and for distinct scenarios.

The question now is how the many parameters defining the context of a service request and acquired from different sources (i.e. user, device and environment profile) can be formally represented, managed and integrated to be used by the upper service layer for adaptation?

Wang et al. [2] identified a set of necessary functional elements that context-aware systems have to support:

- *Context acquisition*, which concerns mechanisms to obtain the context data from different context sources.

- *Context modeling*, which forms the basis for context sharing and interpretation. Current context models differ in the expressiveness they support and the types of context they represent [20].

- *Context Aggregation*: Based on a shared context model, context aggregation merges interrelated information gathered from different sources and enable further data interpretation, and alleviating context-aware services from overhead caused by querying from distributed context sources.

- *Context Interpretation*: The low-level information needs to be interpreted to derive high-level context understood and utilizable by services.

- *Context Query*: Context-aware services need a mechanism to access to interrelated information spread across distributed context repositories.

3.9.2. Characteristics of Context Information

One of the main design considerations of context-aware systems is the operating context representation to capture its features. Such a representation has to be flexible and powerful to allow applications to easily react at provision time to the frequent context changes.

In [8, 19], contextual information resides at three levels:

- The environment level enables defining the overall environment context.

- The service level models and manages the context surrounding individual services offered over mobile devices.

- The resource level presents the context of resources on which the services are to operate.

3.9.3. Context-aware Personalization

Mobile devices enable users access a wide range of services without guiding them through their actual demand. Thus, to provide mobile users with an acceptable and affordable set of services and information the offered set must be custom-tailored to the individual needs. In other word, they must be personalized. Two important categories of personalization motivation can be identified: personalization to facilitate work and personalization to accommodate social requirements.

3.9.4. Personalization Approaches

We can distinguish three approaches to achieve personalization:

- Location-based: meaning that a user's location is taken into consideration for service provision.

- Context-aware: meaning that beyond location information, service provision takes into account user's environment context information. Obviously this helps delivering the right service at the right time.

- Situation-aware: meaning that an abstraction of context information could be done by translating this information into logical situation (e.g., being in a car, in a conference room, eating, etc).

4. Abstract Architecture of Context-aware Systems

As a result of literature review that explores the context-aware systems, we present here general layer architecture for context-aware systems. Figure 1 illustrates this architecture which consists of 3 layers: (i) network layer, (ii) mobile middleware layer, and (iii) application layer.

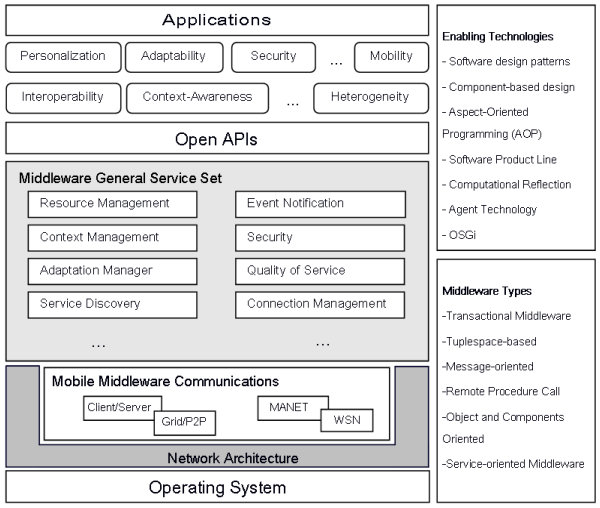

Figure 1. Abstract Architecture of Context-aware Systems

Mobile middleware consists of two parts: generic service set and synchronous and asynchronous communication support. Generic services set include for example, context management, resource management, service discovery, etc. There is no absolute core set of middleware services that all applications require [3].

5. Middleware

5.1 Role of Middleware

Development of ubiquitous applications is a complex and error-prone task because they must cope with heterogeneous infrastructures and with system dynamics in an open network. Middleware role is therefore essential to support mobility and adaptation of applications to the current context [9].

5.2 Middleware Categorization

Middleware platforms can be classified according to the coordination model they implement, as the following [10]:

- *Transactional Middleware*: it offers an interface for running transactions among distributed parties.

- *Tuplespace-based Middleware*: it abstracts distributed tuplespace, which further managed in order to assure non-functional properties like reliability, persistence and scalability.

- *Message-Oriented Middleware*: it provides functionality to publish, select and deliver messages with properties like persistence, real-time performance as well as scalability and security.

- *Remote Procedure Call*: it offers services for: generating client/server stub, marshalling/ unmarshalling data (parameters), establishing synchronous communication as well as assuring non-functional properties.

- *Object and Components Oriented Middleware*: it offer tools that: generate stubs from the object (component) interface specification, obtain a reference to a remote object (component) to interact with, to establish synchronous communication, and invoke requested methods (operations) by marshalling and unmarshalling exchanged data.

- *Service-Oriented Middleware*: it supports the development of distributed software systems in terms of loosely coupled networked services. Service-oriented middleware hides the heterogeneity of the underlying environment by introducing languages for service description and protocols for service discovery and access.

5.3 Context-aware Middleware Requirements:

- *Connection Management*: Due to the frequent disconnection and reconnection of mobile devices Middleware has to provide asynchronous communication. Additionally it has to decouple sender and receiver as well as manage data synchronization.

- *Resource Management*: The limitation of the hardware resources introduces another design constraint on the middleware: Mobile middleware has to be lightweight [3]. On the other hand, it has to efficiently use available resources and avoid overloading the device itself.

- *Context Management*: is responsible for aggregating context information from different sources, storing them in appropriate format, and provide querying and notification support. Mobile middleware must represent context to upper layer and communicate changes until the service layer is reached. It is in this layer the decision takes place about the best way to react to the context change. The middleware should be implemented in such a way as to achieve a trade-off between transparency and awareness [3].

- *Publishing Services*: Middleware should provide a way to publish, in a registry known to all parties, the service description, its context information, and types of contexts it is interested in.

- *Service Discovery*: Middleware should provide the possibility for the client to discover services based on

the service description, its contextual information or the context it is interested in receiving from the service.

- *Adaptation*: Middleware should provide behavior adaptation triggered by this context change.

- *Reliable Asynchronous Invocation:* middleware should support asynchronous invocation so that clients can send request to service when connected and then later receive response during reconnection phase.

6. Developing Context-aware Applications in Pervasive Environment

Designing and implementing distributed applications for such environment is a complex task [16, 28]. In particular, application adaptation based on context such as environmental factors, device limitations and connectivity, requires the programmer to handle a complex combination of factors that manifest themselves throughout the application [6].

6.1 Context-aware Service Discovery

In general, we can distinguish three categories of service discovery and interaction platforms [17]:

- Solutions in which discovery protocols are supported by mobile code. That is, after discovery the service is downloaded onto the mobile device where it can operate. Example of such solutions is Jini.

- Solutions where the discovery protocols are integrated with interaction protocols, which are used to invoke the service after the service has been discovered. Examples are: Universal Plug and Play (UPnP) with SOAP, Saturation with Sun Remote Procedure Call (RPC), and Gaia with Common Object Request Broker Architecture (CORBA).

- Solution where the discovery protocols are independent from interaction protocols. These discovery protocols can be integrated with a range of interaction protocols. Examples are: Service Location Protocol, and ReMMoC [17].

It has been shown that incorporating context and situation awareness in service discovery can greatly improve the precision and recall of the discovery results [3], where recall is defined as number of relevant services retrieved in service discovery divided by the total number of relevant services available, and precision is defined as the number of relevant services retrieved in service discovery divide by the total number of services discovered.

6.2 Context-aware Service Composition

6.2.1 Composition in Mobile Environment

Research in service composition has followed two directions: One direction defines languages to formally describe services and composite services in terms of e.g. service input/output, service constraints and invocation mechanisms. This research also includes developing engines that utilize these languages to generate workflow specifications that compose different services. The other direction concerns development architectures [13] that enable service composition. Based on a declarative description of services, these architectures perform the task of discovering, integration and execution of the relevant services.

Service composition facilities in general, and service composition incorporating personalization and context awareness in particular, offer the potential to simplify the life of the mobile user [1].

6.2.2 On Personalizing Web services Composition

We can identify personalization from two perspectives, each one raises questions:

- The Web content provisioning perspective raises questions like: what content should to deliver to the user, how to ensure user's privacy, and how to create a global personalization schema?

- Web services provisioning perspective raises questions like: at what level can the Web service be personalized, does Web service personalization occur before or after composition, to what extent can a user personalize a Web service without altering its consistency, and does Web service personalization have to adhere to specific policies? [5].

6.3 Context-aware Adaptability

The need for adaptability in pervasive environment is particularly evident at the "wireless edges" of the Internet where the recurring change of operating context (e.g. wireless signal loss), require from the mobile device software to find compromise approaches to the conflicting concerns such as quality-of-service (QoS) and energy consumption [14].

6.3.1 Adaptation Difficulties

The difficulty in developing and maintaining adaptable applications is [14] the cross-cutting nature of the adaptive code. That is, the adaptive code tends to crosscut the functional code. Furthermore, developing new adaptable application is more challenging than enhancing existing ones by dynamic aspects not considered during their design.

6.3.2 Adaptation Strategies

Context-aware systems have the ability to seamlessly adapt their behavior according to the environment context. This adaptation may take several forms, spanning: changing internal processing, altering

the context being processed and exchanged over the network, and modifying user interfaces [10].

Looking at pervasive application development from the software engineering perspective, we observe that the environment state, device limitations, user mobility and connectivity have an impact across the application as a whole (in other words, they are crosscutting).

7. Conclusion

The potential advantage gained from the cooperation the services discovery and composition from one side and the personalization and context-awareness form the other side has two advantages. First, it provides a focused composed service and tailored to user needs and wants, and second, it alleviating the user from searching for services most appropriate for their needs among huge amount number of available services.

Therefore, by making use of a close coupling between discovery and composition, personalization and context awareness using Service-oriented Architecture (SOA) mechanism to create a truly user-centric environment could be established.

Finally, given the heterogeneity, mobility and adaptation aspects in the distributed computing environment, SOA could be a good candidate since it offers seamless integration between wired and unwired environments.

8. References

[1] A. Davy , F. Mahon , K. Doolin , B. Jennings , M. Foghlú. Secure Mobile Services Infrastructures for mGovernment: Personalised, Context-aware Composition of Pervasive Mobile Services. *1st Euro Conf. on Mobile Government, Euro mGov 2005*, Brighton, UK, 2005.

[2] W. Xiaohang. The Context Gateway: A Pervasive Computing Infrastructure for Context Aware Service. *Research Report, School of Computing, National University of Singapore & Context -Aware Dept., Institute for Infocomm Research*, November, 2003.

[3] P. Bellavista, A. Corradi. *The Handbook of Mobile Middleware. Auerbach Publications,* 2006.

[4] K. Boukadi et al. CWSC4EC: How to Employ Context, Web Service, and Community in Enterprise Collaboration. *NOTERE '08: Proceedings of the 8th int conference on New technologies in distributed systems*, 2008.

[5] Z. Maamar. On coordinating personalized composite web services, *Information and Software Technology* 48 (2006) 540–548, ELSEVIER.

[6] A. Carton et al. Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing. *SEPCASE'07: Proceedings of the 1st Int Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, IEEE Computer Society, 2007.

[7] S. M. Sadjadi et al. Transparent Shaping of Existing Software to Support Pervasive and Autonomic Computing. *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, ACM, 2005.

[8] M. Weiser, "The computer for the twenty-first century," *Scientific American*, pp. 94-104, 1991.

[9] H. Schmidt, F. J. Hauck. SAMProc: Middleware for Self-adaptive Mobile Processes in Heterogeneous Ubiquitous Environments. *MDS '07: Proceedings of the 4th on Middleware doctoral symposium*, November 2007

[10] V. Issarny, M. Caporuscio, N. Georgantas. A Perspective on the Future of Middleware-based Software Engineering. IEEE Computer Society, *FOSE '07: 2007 Future of Software Engineering,* May 2007

[11] P. Inverardi, F. Mancinelli, and M. Nesi. A declarative framework for adaptable applications in heterogeneous environments. *In Proceedings of the 19th ACM Symposium on Applied Computing*, 2004.

[12] R. C. Rocha, M. Endler, T. S. Siqueira. Middleware for Ubiquitous Context-Awareness. *MPAC '08: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*, December 2008.

[13] K. Geebelen et al. Dynamic Reconfiguration Using Template Based Web Service Composition. *MW4SOC '08: Proceedings of the 3rd workshop on Middleware for service oriented computing*, ACM, December 2008

[14] S. Masoud Sadjadi, Philip K. McKinley, Betty H.C. Cheng. Transparent Shaping of Existing Software to Support Pervasive and Autonomic Computing. *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, ACM, July 2005.

[15] E. Niemelä , J. Latvakoski. Survey of Requirements and Solutions for Ubiquitous Software. *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, ACM, October 2004.

[16] G. Edwards et al. Self-* Software Architectures and Component Middleware in Pervasive Environments. *MPAC '07: Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing*: held at the ACM/IFIP/USENIX 8th International Middleware Conference, ACM, November 2007.

[17] P. Grace, G. S. Blair, S. Samuel. A Reflective Framework for Discovery and Interaction in Heterogeneous Mobile Environments. *SIGMOBILE Mobile Computing and Communications Review* , Volume 9 Issue 1, ACM, 2005.

[18] Tomasz Rybicki. Semantic Service Discovery in Pervasive Computing Environment. *ICPS '08: Proceedings of the 5th international conference on Pervasive services*, 2008.

[19] Umesh Bellur, Siddharth Bondre. Towards Seamless User Mobility in Service Oriented Environments Via Context Awareness. *ICPS '08: Proceedings of the 5th international conference on Pervasive services*, July 2008.

[20] Abdelghani Chibani et al. Semantic Middleware for Context Services Composition in Ubiquitous Computing. *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, ICST February 2008.