Aspect-Oriented Reconfigurable Middleware for Pervasive Systems

Gustavo G. Pascual

Departamento de Lenguajes y Ciencias de la Computación
University of Málaga, Málaga (SPAIN)
CAOSD group (http:/caosd.lcc.uma.es), gustavo@lcc.uma.es

Abstract. One of the main features of pervasive computing systems is their need to be dynamically reconfigured in order to properly adapt to the continuous changes in their environment conditions (context). An appropriate solution to provide reconfigurability is Aspect-Oriented Software Development, which allows that optional functionalities can be enabled/disabled and services can be replaced with lighter implementations which are less resource consuming. The primary aim of this work is the definition and implementation of an aspect-oriented middleware architecture to dynamically reconfigure pervasive computing systems. Our middleware will provide support for the *context-aware*, *efficient*, *consistent* and *homogeneous* reconfiguration of middleware and application services. Other requirements of pervasive computing systems such as *mobility*, *fault tolerance* and *service distribution* will also be addressed.

Keywords: Middleware, AOSD, Dynamic Reconfiguration, Evaluation

1 Introduction

Nowadays pervasive systems have acquired special relevance due to the popularity of smartphones and embedded systems. In these kinds of systems, the context and, consequently, the requirements that services have to satisfy, change continuously. Therefore, context-aware adaptation is needed, which involves enabling, disabling and modifying services. Moreover, adaptation should be as transparent as possible to the services and the applications that use them. Since performing adaptation is a complex task, it should be addressed by a middleware architecture that provides the monitoring and reconfiguration services needed to achieve context-aware adaptation at both the infrastructure and the application services.

Finding solutions to this problem is not easy since we need to face some significant difficulties. For instance, in these kinds of systems we can find devices with very different capabilities with respect to processing, memory and connectivity resources, so applications and middleware should be reconfigured deploying lighter or more complete versions of their services without affecting the global functionality of the system. Furthermore, whenever reconfiguration is performed it is possible that the new deployed configuration is not consistent. However, controlling consistency introduces a significant overhead, so it is necessary to find a balance between reconfiguration flexibility

and performance. Finally, the services can be distributed, so many non-functional requirements like security, privacy, fault tolerance or mobility need to be addressed. The main motivation of our work is that although the importance of these requirements in the development of pervasive systems has been identified in the literature, no single existing proposal satisfies them all.

Aspect-Oriented Software Development (AOSD) is an appropriate technology for this purpose. With AOSD, optional functionalities can be modelled as aspects, which are dynamically enabled/disabled or even woven/unwoven if the underlying AO platform supports it [1]. Furthermore, services components can also be replaced by alternative implementations that better suit the current context conditions. The final goal of our research is to define an Aspect-Oriented (AO) reconfigurable middleware architecture that perform context monitoring and reconfigure itself in order to adapt to the changes in the environment at both the middleware infrastructure and the application level, while addressing all the requirements previously mentioned.

2 Related Work

Table 1 summarises the requirements that have been identified in pervasive computing systems. For each requirement, it indicates which proposals address it. Next, we can find more details about these requirements.

Proposal	**R**	**C**	**P**	**D**	**M**	**FT**
MobiPADS [2]	Application	No	No	Yes	No	No
RCSM [3]	Application, Monitoring Service	No	No	No	Yes	No
SOCAM [4]	Monitoring Service	No	No	No	Yes	No
MAUI [5]	Application	No	No	Yes	No	Yes
Calling the Cloud [6]	Application	No	No	Yes	No	No
Adaptive Online Deployment... [7]	Application	No	No	Yes	No	No
Janik et al. [8]	Application, Monitoring Service	No	No	Yes	Yes	No
CoDAMoS [9]	Application	No	No	Yes	No	No
Paspallis [10]	Monitoring Service	No	No	No	No	No

R:Reconfiguration **C**:Consistency **P**:Privacy **D**:Distribution **M**:Mobility **FT**:Fault Tolerance

Table 1. Comparison of requirements fulfillment of related work

As has been previously mentioned, reconfiguration is necessary because the environment and the available resources are continuously changing. Reconfiguration can be provided for applications, middleware services or both of them. Some work [2, 5, 6, 9] provide only reconfiguration of application layer. In these proposals, the amount of resources that the middleware takes is always the same, so these approaches are not suitable for pervasive computing systems. Other work such as RCSM [3], SOCAM [4], the work by Janik and Zielinski [8] and the architecture in the PhD thesis of Paspallis [10] provide reconfiguration of middleware layer. However, as reconfiguration is

mainly provided for context monitoring service, this requirement is only partially addressed. RCSM, SOCAM, and the work presented in the thesis of Paspallis are able to enable and disable sensing units (i.e. context providers) at runtime. The work of Janik and Zielinski integrates with AAOP, an AO model which provides support for specifying adaptability strategies through a graph. With AAOP, sampling devices can be enabled/disabled and applications can be reconfigured in order to satisfy non-functional requirements. However, this approach lacks flexibility because the number of states of the graph is multiplied with every non-functional requirement taken into account.

In these work, only small changes are performed during reconfiguration, so there is no need to provide mechanisms to ensure consistency. In some proposals that provide dynamic reconfiguration (e.g. [11–14]) consistency is not addressed, or it is only partially addressed.

Reconfiguration is a mechanism that provides support for adaptation to changes in the context of the system. Therefore, one of the most relevant services is context monitoring. Most approaches acquire context data in a similar way, and the only significant difference is introduced by aspect-oriented approaches. Concretely, in Janik and Zielinski, context information is provided by sampling devices that are implemented as aspects and can be woven/unwoven as needed. Some kind of context abstraction, like data aggregation or filtering, should be provided. RCSM provides support for past data aggregation introducing the concept of *situation*, and the proposal by Janik and Zielinski follows a similar approach. On the other hand, SOCAM uses an Ontology Web Language (OWL) and in the work by Paspallis an XML-based language called CQL is used to access context information. None of them is adaptable enough in order to satisfy the necessities of all context-aware applications.

Frequently, the context data sources are remotely instantiated, so the middleware should provide support for accessing remote context. Furthermore, since some kinds of context information (e.g. location) are especially sensitive, the middleware should provide mechanisms to ensure that only authorised entities can access that information. Therefore, two new requirements (privacy and security) arise. All proposals provide support for accessing remote context but none of them address privacy or security.

Access to remote context data could be provided through a distributed context monitoring service. However, it would be desirable that not only context monitoring but every service could benefit from this feature. Distributed services can be remotely instantiated, enabling middleware to run services in devices with more available resources (i.e. computing resources, battery, etc.). In the related work presented in this section, distribution of middleware layer is only provided for context monitoring service.

Although reconfiguration is an important requirement in pervasive computing systems, more requirements have been identified. One common feature in pervasive computing systems is mobility. In mobility situations, the reachability of devices and services changes frequently, and some mechanisms are needed in order to handle these cases. In RCSM, the protocol R-CDP provides support for looking for new sensors dynamically, and SOCAM provides a *Service Location* service in which context providers are registered. On the other hand, mobile devices are prone to failures, so fault-tolerance mechanisms should be provided too. However, these proposals do not address this requirement.

In this section we have focused on research that provides support for reconfiguration. It is worth noting that we can find other work such as *The Context Toolkit* [15] and PACE [16] in which reconfiguration support is not provided but some of the mentioned requirements are addressed. The Context Toolkit provides support for privacy since context providers can be assigned an owner and a set of rules for context sharing. Mobility is also addressed through a component discovery mechanism. PACE middleware provides support for privacy, mobility and fault tolerance. Privacy is performed with access control in context repositories, mobility support is provided through a publish/subscribe mechanism, but fault tolerance is only partially addressed since it is only provided for sensing errors or failed sensors, but not failed components.

3 Proposed Solution

With our proposal we provide two main contributions. The first one is a middleware architecture that provides support for a homogeneous reconfiguration of middleware and application services. Second, we propose a service model that covers all the stages of the software development lifecycle, providing support for specifying and developing services that can take advantage of the reconfiguration service.

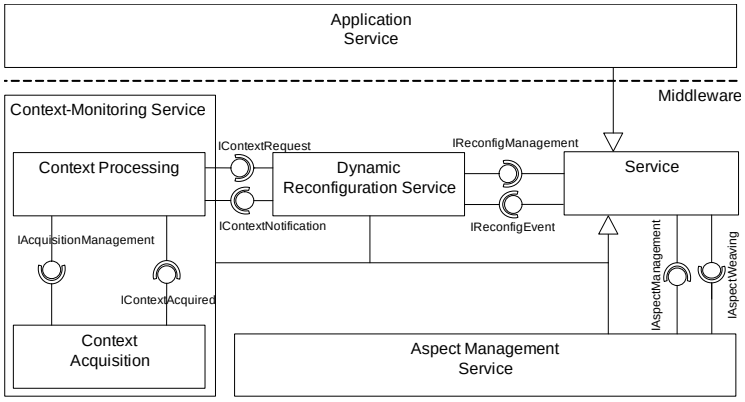

Fig. 1. Middleware Architecture

Figure 1 shows our high level middleware architecture. In this Figure, the Dynamic Reconfiguration Service (DRS), the Aspect Management Service (AMS) and the Context Monitoring Service (CMS) are explicitly shown because these services are critical parts of our middleware architecture. DRS provides support for services reconfiguration, and takes into account the context information provided by CMS. DRS and CMS provide a well known interface and do not depend on services internal architecture, so every application and middleware service can take advantage of the dynamic reconfiguration and the context monitoring services in a homogeneous way. In our architecture, aspects will be woven and unwoven dynamically by AMS. In this way, new functional-

ities can be added or removed from services according to previously defined reconfiguration policies.

Figure 2 describes the process of services development and reconfiguration. Development of a service begins with the specification of a features model. This model includes mandatory and optional features of the service, that will be dynamically enabled and disabled based on context data. From the features model and context data a repository of configurations will be defined, which consists of a set of valid and consistent architectures, each one addressing a valid combination of features of the model. At runtime, the DRS will dynamically select the appropriate configuration according to the current context.

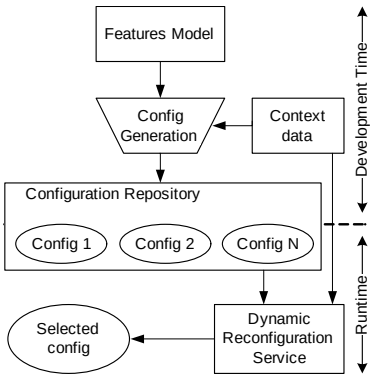

Fig. 2. Service Modelling

4 Research Methodology

Our objective is to define a middleware architecture in which every infrastructure and application service can be reconfigured in a homogeneous way while fulfilling typical requirements of pervasive systems, and we will achieve this objective ensuring that low overhead is introduced.

In the process of services specification, all the steps of software development life cycle will be followed. We adopt the same principles of the Software Product Lines (SPL) [17] approach. However, instead of defining a family of products we will define a familiy of valid architectures for every service. In the process of specifying a service, the first step is to define its features model. This model will include both mandatory and optional features of the service and the dependences between these features. Every valid choice of features of the model will be maped to a specific architecture of the service, which will include both base and aspectual components. In order to improve this process, technologies like *Model Driven Development* [18] could be applied.

One of our main objectives is that our middleware can be deployed in resource constrained devices. For this reason, we will design and evaluate the most critical parts

(in terms of resources consumption) of the system first. Once we are sure that the critical parts of the system are designed in an efficient way, we will define the rest of the services of the architecture. One of these parts is the Aspect Management Service. Since this service provides mechanisms to instrument the communication between components, the design and implementation of other services of the middleware will depend on the design of AMS. Another critical part of the architecture is the Dynamic Reconfiguration Service, so it is necessary to provide mechanims to perform reconfiguration (enabling and disabling features from the service model) from context monitoring data in a very efficient way. Additionally, this service should provide support for defining bindings between context information and reconfiguration triggering.

5 Expected Contribution and Current Status

Through the proposed middleware architecture and methodology four contributions are expected.

1. *Homogeneous services reconfiguration*. We propose an architecture in which both application and infrastructure services can be reconfigured in a homogeneous way. In related work, reconfiguration is frequently limited to application layer and, if middleware is reconfigurable, it is restricted specifically to some services.
2. *Extensible architecture*. Since services can be developed independently, the middleware architecture can be extended with new services as needed. This way, both functional and non-functional requirements can be fulfilled for any use case.
3. *Consistent and efficient reconfiguration mechanism*. All the valid configurations of a service are defined at design time in its features model and associated service architectures. This way, consistency is guaranteed without introducing additional overhead. An initial efficiency evaluation of the weaving mechanism has been performed in [19], and reconfiguration performance is being evaluated in ongoing work.
4. *Addressing pervasive computing requirements*. Our work, unlike the related proposals, will address pervasive computing requirements like privacy, security, service distribution, mobility and fault tolerance.

Currently, we have specified the context monitoring service and have implemented and evaluated a prototype of the Aspect Management Service, which is one the critical parts of the system. A first evaluation was presented in [19], although these results have been significantly improved in ongoing work.

The next service to be defined in detail is Dynamic Reconfiguration Service, another key part of our middleware architecture.

6 Evaluation Plan

Each time a newservice is introduced into the middleware infrastructure or modified, an evaluation has to be performed in order to ensure that the service works properly and, since we target resource-constrained devices, that it works in an efficient way. To this

end, appropriate scenarios will be defined. The results obtained will be compared, when possible, with the results obtained from related work.

First, critical parts of the system will be evaluated, since they affect directly to the rest of the system. In this way, the first service that has been evaluated is Aspect Management Service. To this end, we have evaluated AO general-purpose languages that provide dynamic reconfiguration, such as JBoss AOP; other AO languages (e.g. Spring AOP) will also be evaluated as part of our future work. However, since these languages need to address the requirements of every kind of application, our initial results show that they introduce a high overhead and, consequently, they seem not suitable for resource-constrained devices. Therefore, we have implemented and evaluated a prototype of this service that meets the needs of our architecture. In [19], performance in the composition process was evaluated, which is very important since one of the main drawbacks of AOSD is the overhead that dynamic composition introduces. Furthermore, since according to an AO approach, reconfiguration would normally involve not only changing parameters but also adding and removing components, it is also highly important to evaluate the reconfiguration performance of our approach. As part of our ongoing work, reconfiguration performance is being evaluated too, and other requirements such as consistency will also be addressed.

Once we have developed the main services of the middleware architecture, some application scenarios will be defined and implemented. In these scenarios, different kinds of devices such as smartphones, laptops and desktop computers will be used, and both application and middleware services will be reconfigured according to context information provided by sensors (e.g. location, temperature), the operating system (e.g. CPU and memory consumption) or the user (e.g. user preferences). These applications will involve requirements such as mobility, distribution and fault tolerance while taking into account the resource-constraints of these kinds of devices.

Acknowledgements This research has been conducted in collaboration with Mónica Pinto and Lidia Fuentes from the Languages and Computer Science Department at the University of Málaga, which are the PhD Supervisors of the thesis discussed in this paper.

References

1. JBoss-AOP: (http://www.jboss.org/jbossaop) Last visited: February, 2011.
2. Chan, A., Chuang, S.: MobiPADS: a reflective middleware for context-aware mobile computing. IEEE Transactions on Software Engineering (2003) 1072–1085
3. Yau, S.S., Huang, D., Gong, H., Seth, S.: Development and runtime support for situation-aware application software in ubiquitous computing environments. In: In COMPSAC, IEEE (2004) 452–457
4. Gu, T., Pung, H., Zhang, D.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications **28** (2005) 1–18
5. Cuervo, E., et. al: Maui: Making smartphones last longer with code offload. In: Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM (2010) 49–62

6. Giurgiu, I., et. al.: Calling the cloud: enabling mobile phones as interfaces to cloud applications. In: Proceedings of the 10th International Conference on Middleware, Springer-Verlag (2009) 1–20
7. Verbelen, T., Hens, R., Stevens, T., De Turck, F., Dhoedt, B.: Adaptive online deployment for resource constrained mobile smart clients. In: MOBILe Wireless MiddleWARE, Operating Systems, and Applications, 3rd International ICST conference, Proceedings. (2010)
8. Janik, A., Zielinski, K.: AAOP-based dynamically reconfigurable monitoring system. Information and Software Technology **52** (2010) 380–396
9. Team, C.D.: Codamos: Context-driven adaptation of mobile services. (`http://distrinet.cs.kuleuven.be/projects/CoDAMoS/`)
10. Paspallis, N.: Middleware-based development of context-aware applications with reusable components. University of Cyprus (2009)
11. Grace, P., et. al.: Deep middleware for the divergent grid. In: Proceedings of the International Conference on Middleware, Springer-Verlag (2005) 334–353
12. Hillman, J., Warren, I.: An open framework for dynamic reconfiguration. In: Proceedings of the 26th ICSE, IEEE Computer Society (2004) 594–603
13. Rasche, A., Polze, A.: Configuration and dynamic reconfiguration of component-based applications with microsoft. net. (2003)
14. Vandewoude, Y.: Dynamically updating component-oriented systems. status: published (2007)
15. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. **16** (2001) 97–166
16. Henricksen, K., Indulska, J., McFadden, T., Balasubramaniam, S.: Middleware for distributed context-aware systems. On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE (2005) 846–863
17. Pohl, K., Böckle, G., der Linden, F.J.V.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer (2005)
18. Beydeda, S., Book, M., Volker, G.: Model-driven software development. Springer, City (2005)
19. G. Pascual, G., Fuentes, L., Pinto, M.: Towards a Reconfigurable Middleware Architecture for Pervasive Computing Systems. In: 9th International Workshop on System/Software Architectures. (2011) Accepted for publication.