Multi-dynamics adaptations using Cascaded Aspect of Assembly

Nicolas Ferry
I3S (UNS - CNRS) and CSTB
930 Route des Colles - BP 145
06901 Sophia-Antipolis France
nicolas.ferry@unice.fr

Stéphane Lavirotte
I3S (UNS - CNRS)
930 Route des Colles - BP 145
06903 Sophia-Antipolis France
stephane.lavirotte@unice.fr

Jean-Yves Tigli
I3S (UNS - CNRS)
930 Route des Colles - BP 145
06903 Sophia-Antipolis France
tigli@unice.fr

Gaëtan Rey
I3S (UNS - CNRS)
930 Route des Colles - BP 145
06903 Sophia-Antipolis France
gaetan.rey@unice.fr

Michel Riveill
I3S (UNS - CNRS)
930 Route des Colles - BP 145
06903 Sophia-Antipolis France
riveill@unice.fr

ABSTRACT

Middleware for ubiquitous computing are having a mediator role in ubiquitous systems. They have to be able to handle both dynamics: the dynamic of the environment and of the application. But, at least, they must consider the dynamic of the underlying software infrastructure. To handle such a dynamic, we present in this paper a mechanism for adaptation called cascaded aspects of assembly (AA). The functional decomposition proposed by cascaded AAs allows to dynamically compose adaptations to manage the dynamic variability of the application and of its infrastructure. Because cascaded AA can be selected/unselected or edited at runtime, some others processes can manage them in various autonomous and decentralized processes to handle the other dynamics.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Configuration Management

Keywords

Ubiquitous computing, software composition, aspect-oriented programming, context-awareness

1. INTRODUCTION

Nowadays, with the miniaturization of computer hardware, many objects with computational capabilities are dissolving in our daily life. A ubiquitous system is a sphere of interactions of such objects also called devices. To create an editable and then adaptable software application, the system must enable and manage (compose) interactions between these devices. This infrastructure is highly dynamic due to arbitrary node mobility. Because of this variability and because the environment is continuously evolving, ubiquitous systems have to handle those changes. They must be able to adapt to their context. Context awareness including adaptation [2] is a crosscutting concern [3]. Mechanism to address this concern must then be proposed by middleware for ubiquitous computing. Such middleware are playing a mediator role since they are the link between the environment and the application (and the user through these two mediums). This role implies that such a middleware have to be able to handle both dynamics: the dynamic of the environment and of the application. To address these issues, we propose in this paper an approach to handle these dynamics in a decentralized way using various autonomous processes. Each process is pursuing its own dynamic.

ARM '2010, November 30, 2010, Bangalore, India.

2. MULTI-DYNAMICS MIDDLEWARE

From a same environment or application, the dynamics previously described can be considered in various ways depending on the concern addressed. But, at least, it must consider one dynamic: the dynamic of its underlying software infrastructure. We can talk of *physical dynamic*. Indeed, when a device disappears the system should no longer use it. This may lead to the production of a nonworking application. Similarly, the created or adapted system should not use a missing service.

Once this pre-requisite is ensured, some other dynamics can be considered. This may consists in considering the context using a logical approach in order to identify some adaptations that must be done according to a situation. Whatever the mechanism implemented for this, it must always consider the physical dynamic of the environment. This must, at least in the case of a device disappearance, be considered as fast as possible. However, in a logical approach the set of data that must be considered is potentially huge. This may not allows reacting in time to an infrastructure evolution. This means that the adaptation mechanism have to be scalable and reactive. **So logical and physical dynamics must be considered separately and the physical dynamic have to be respected for each adaptation.** The mechanism to handle the physical dynamic has to be strongly associated to the adaptation mechanism. The

various mechanisms to consider a dynamic based on a logical approach can be seen as a way to manage mechanism for adaptation and the physical dynamic. So, first layers are to adapt the system in an opportunistic way according to the underlying infrastructure. Others layers are to manage adaptations according to the context to produce a global coherent behavior to the system. We will now present an implementation of the adaptation mechanism for such an approach and how it can handle the infrastructure dynamic. Then we will see how it allows to handle the various dynamics previously described using separated processes.

3. PHYSICAL DYNAMIC AND ADAPTATION USING CASCADED AA

Aspects of Assembly (AA) [4] are based on the AOP model to define adaptation schemas and on a weaving process with logical merging. They allow structural reconfiguration of components assemblies at runtime, keeping black-box property of components. Modifications include adding components and bindings between them. An advice describes a structural reconfiguration of a components assembly, while a pointcut identifies components' ports (joinpoints) on which changes will take place. Pointcut describes a part of the physical environment. The pointcut matching will then be the mechanism to manage the physical dynamic. Originally an AA is in an unselected state. This means that the user does not want to apply it. In such a case, its pointcuts are not even evaluated. When an AA is selected, its pointcuts are evaluated. They will be evaluated for each modification of the assembly on which will be applied the AA. If some joinpoints are satisfying all the pointcut rules of an AA, it becomes relevant before being woven. Similarly, those that have been woven can become unapplied and still selected if an entity identified by their pointcut disappears. So, AA's weaving is in line with the evolution of the infrastructure.

A cascade of AA consists in chaining several weaving process. So a component instantiated by an AA in the assembly may trigger the weaving of another AA. A concern can be expressed as an ordered set of unordered sets (called group afterward) of AA . Each group of AA is specified for a specific process of weaving. The group of rank n will be woven during the weaving process number n. The functionalities implemented in a weaving process can be reused for various concerns. Moreover, according to the underlying infrastructure, AA from a group will be dynamically composed with other AA from other groups. It allows to define a maximum number of system's configurations from a reduced set of AA and to manage the dynamic variability of the application. So, according to their infrastructure, AA from cascaded AA are woven and composed together.

4. LOGICAL DYNAMICS

A weaving process can be triggered by changing, at runtime, the set of AA given as input to the weaver. By selecting/deselecting or adding/removing aspects of assembly at runtime. When the set of AA is modified, the weaver is triggered; leading to adaptation if an added AA can be applied or if an AA has been removed.

Then, with a dynamic of its own, another entity can trigger and manage a set of adaptation schemas, a set of cascaded AA. This set of selected and deployed cascaded AA will also pursue its own dynamic with respect to the physical dynamic. **They are autonomous and separated process. So, while the mechanism that logically evaluates the context is processing, the system is still able to react to changes in its software infrastructure.** Moreover, there can be as many autonomous processes as there are various behaviors or concerns in the system. Each of them pursues its own dynamic. This improves system's continuity of service and self-healing capabilities. In this paper we will not present a logical mechanism. But it can consist in some basic event-condition-action rules or using some feature models as in [1].

Thanks to this mechanism, and combined to the weaver's merging engine, we can manage sets of cascaded AA to ensure a consistent global behavior to the system. Moreover, the modularity introduced into cascades could allow composing at runtime various AA from cascaded AA. And then it could allow to create new cascades or to add some constraints on their combination. As an example, we can express some constraints as: if *AA1* and *AA2* can be applied select *AA2* or it can consist in exchanging an AA with another one according to the situation. According to the system goal, the logical process should be implemented either into the application either into the mechanism to manage adaptations. This is up to the designer of the ubiquitous system.

5. CONCLUSION

In this paper we presented how the cascaded AA mechanism allows implementing multi-dynamics middleware. Cascaded AA is a mechanism to build applications in an opportunistic way. Closely related to adaptation, it embeds a mechanism to consider the dynamics of the software infrastructure. So that, the adaptation process always respects this dynamic. Moreover, cascaded AA can be triggered in a logical way. So, we can add to this many processes to consider the context in a logical way for some specific concerns.

6. ACKNOWLEDGMENTS

This work is part of the Continuum Project (French National Research Agency) ANR-08-VERS-005.

7. REFERENCES

[1] M. Acher, P. Collet, F. Fleurey, P. Lahire, S. Moisan, and J.-P. Rigault. Modeling Context and Dynamic Adaptations with Feature Models. In *4th International Workshop Models@run.time*, page 10, Oct. 2009.

[2] A. Bottaro, J. Bourcier, C. Escoffier, and P. Lalanda. Context-aware service composition in a home control gateway. *International Conference on Pervasive Services*, pages 223–231, 2007.

[3] P.-C. David and T. Ledoux. An aspect-oriented approach for developing self-adaptive Fractal components. In *5th International Symposium on Software Composition*. Springer-Verlag, Mar. 2006.

[4] J.-Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, D. Cheung-Foo-Wo, E. Callegari, and M. Riveill. WComp Middleware for Ubiquitous Computing: Aspects and Composite Event-based Web Services. *Annals of Telecommunications (AoT)*, Apr. 2009.