Electronic Communications of the EASST
Volume 11 (2008)

Proceedings of the First International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2008)

Model-Driven Adaptation of Ubiquitous Applications

Carlos Parra, Laurence Duchien

6 pages

Guest Editors: Romain Rouvoy, Mauro Caporuscio, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Model-Driven Adaptation of Ubiquitous Applications

Carlos Parra, Laurence Duchien

University of Lille 1, INRIA Lille - Nord Europe, Laboratoire LIFL UMR CNRS 8022
Parc Scientifique de la Haute-Borne
40, Avenue Halley - 59650 Villeneuve D'Ascq - France
{carlos.parra, laurence.duchien}@inria.fr

Abstract: In ubiquitous computing, applications are executed in powerful and mobile devices allowing users to enter and leave frequently from diverse locations. Mobility becomes the rule rather than the exception. Software should be able to change its behavior and react to changes on its environment. In this proposal paper we discuss several challenges for context-aware software development and present our proposal to achieve software adaptation based on context information. We use a model of context as a starting point for a process where components are generated, deployed, and loaded on mobile devices.

Keywords: Context-awareness, adaptation, model-driven engineering, software components.

1 Introduction

Mobility achieved through cutting-edge devices allows users to enter and leave environments in which they interact with each other. Applications need to be aware and able to react to context changes. By context we mean all the environmental information that may affect the behavior of an application running in a mobile device. This information may include network updates, services availability, user preferences and data received from sensors.

One of the main concerns in the development of context-aware software is to find a way to represent context information. One way to do that is by utilizing models. In the Model-driven Engineering (MDE) [MM03] emphasis is made on raising the level of abstraction in software development, by separating the business elements from the specific details related to platform and implementation issues. Starting from platform independent models (PIMs), mappings towards platform specific models (PSMs) are constructed and then, it is possible to create software assets like code and configuration files. Here, models can be used not only as a way to standardize communication of context information between heterogeneous systems, but also as core assets used in transformation processes to produce source code.

Besides achieving communication, context-aware software should be able to adapt itself. Currently, adaptation is considered as a first-class problem that must be taken into account throughout the software life-cycle. Even further, in ubiquitous computing, a set of challenges associated to restrictions in the execution environment of applications enforces the idea of adaptation.

In this proposal paper we introduce our approach to achieve context-aware software adaptation. We present a scheme in which a server side, which provide a set of services, is in charge of generating several software components based on context updates. The components are sent and

loaded in the client side. Such components add or modify client's functionality and also, enable the client to use services provided by its environment. We start from a model of context, which is used for two main purposes: to exchange context information between devices, and to allow the generation of components that will enrich client applications.

The reminder of this paper is organized as follows. In section 2 we briefly introduce the CAPPUCINO project. In section 3 we present our proposal to achieve context-aware software adaptation and we briefly describe each of their components. Section 4 presents the related work in the area, and finally, in section 5 we point out some conclusions.

2 CAPPUCINO

This paper deals within a funded project, called CAPPUCINO [Cap08]. The purpose of this project is to undertake the design, the deployment and the execution of software in an open, mobile and ubiquitous environment which requires adaptation throughout the lifecycle of the software, this by taking into account the heterogeneity of the devices on which the software will be executed. In the scope of the CAPPUCINO project, we envisage an *open environment*, in which users enter and leave environments that provide a series of services unknown to them. We consider that the place where our project will be deployed is equipped with servers and a wireless network. This lead us to consider a *client/server* rather than *peer to peer* scheme. We focus on utilizing context information to trigger a process of adaptation in the client side, and thus, to be able to communicate and to take advantage of the services provided by its current environment. Each mobile device is able to load bytes by using a communication protocol like `ftp` or `http`. Deployment approaches like the one described in [FGM05] can be used to load the initial components with middleware functionality for adaptation.

3 Proposal

3.1 Introducing Adaptation

We propose an adaptation process divided in two main phases: the loading of a bootstrap application, and the context-aware adaptation.

3.1.1 Loading of a bootstrap application

The Figure 1 shows the process that takes place when a user enters in the coverage area of a server. Before entering this area, the user has a mobile device that we call client, in which is possible to load bytes. This means that the client is only a machine that can communicate with the server but that is not able to use the services provided by the environment or exchange context information.

The server is in charge of getting the hardware capabilities of the client. One approach used in `WEP` developement is through a configuration file like `WURFL` *(Wireless Universal Resource File)* [Pas08]. `WURFL` intends to provide the hardware specifications and capabilities of an increasing number of mobile models. In our proposal we assume that we can stablish a first communication

with the mobile and we are able to get its characteristics by querying a repository like WURFL. Using this information, the server generates a bootstrap application (1). This bootstrap is responsible for three main tasks: to communicate with the environment, to handle context information, and to allow the loading of new components if needed.

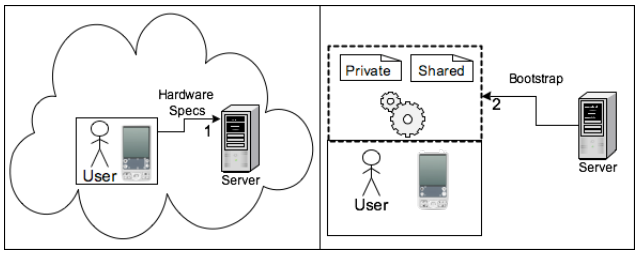

Figure 1: Bootstrap loading.

Once the client has loaded the bootstrap, it creates an instance of its own context information that conforms to the context model provided by the server (2). This information is separated in two parts: a private part to hold personal data and preferences, and a shared part to hold information relevant to the client and to the server.

3.1.2 Context-Aware Adaptation

After the bootstrap has been loaded, the client is ready to interact with its environment. In the second phase, an adaptation occurs if the client wants to use new services. For this process of adaptation we define four main steps as shown in Figure 2. The client's bootstrap sends its shared context information (1). The server uses it to generate one or more components with the funcionality required by the client (2). Then, these components are sent to the client together with relevant context information (3). The client loads the new components, and using private context, filters the incoming information to match user preferences (4).

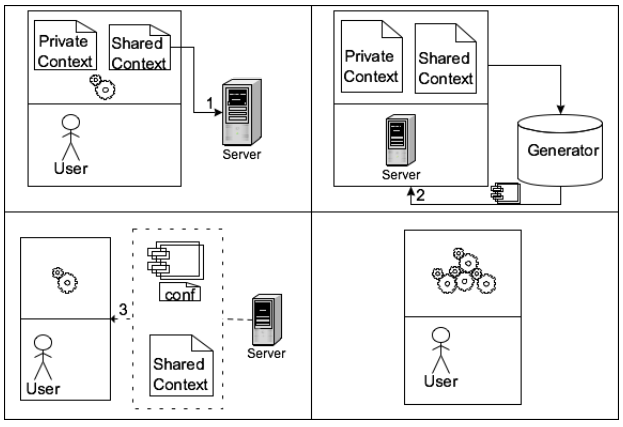

Figure 2: Context-aware adaptation process.

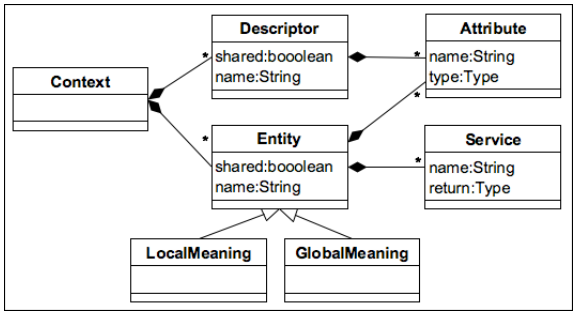

Figure 3: Context Metamodel.

3.2 Model-based generation

In our approach, we intend to generate dynamically both the bootstrap and the components. To achieve this, we center the generation process around a context model and a set of metamodel transformations. The process begins with a context model that conforms to a context metamodel and is transformed first into an application model, and later into code. Our context metamodel is shown in Figure 3). It separates context information in entities and descriptors and introduces the notion of global and local context to determine which information is always valid and which makes sense only in specific areas. The private information (as client preferences) is used by the bootstrap, together with the shared information, to provide customized services to the client.

The `Context` element refers to the root of the context information. The `Context` represents a space in which there are a set of `Entity` elements and information represented by `Descriptor` elements. A `Descriptor` represents a piece of context information. For example, a `Descriptor` may contain the current time, a set of coordinates for a given location, the temperature of a room, etc. An `Attribute` represents a property of a `Descriptor` or an `Entity`. An `Entity` represents a participant in a given environment, that is, an element that interacts with each other by offering and requesting `Services`. Examples of entities vary from devices as computers, mobile phones and PDA's to users and software applications. The attribute `shared` in both `Entity` and `Descriptor` elements is intended to make the distinction between private and shared information. The classes `GlobalMeaning` and `LocalMeaning` separate the `Entity` elements in two groups. `LocalMeaning` elements represent information which meaning is restricted to a particular place or context, for example a restaurant menu. On the other side the `GlobalMeaning` represents information that remains the same in different environments, an example of such information could be the operating system of a mobile phone or the number of pixels of its screen.

4 Related Work

There are several topics found in the literature that deal with context-aware software development and adaptation. A close-related area to our proposal is *Component-Based Software Engineering* [Szy98] and specially adaptation using components. Applications can be adapted by adding or

removing components statically (at compile time) or dynamically (at runtime)[PSKC04]. We plan to introduce component adaptation, nevertheless, in our approach we start with a generation process in which we produce new components and configuration files from a context model and then, these generated components can be deployed and loaded dynamically using a component-based platform. Context modeling has ben studied in several approaches. [SP04] presents a classification of context models in six groups. *Key value models*, *markup schemes*. *graphical models*, *ontologies*, *object oriented models* and *logic models*. Our metamodel can be classified in the graphical models group since we use a class diagram to represent it, and also can be classified as an object oriented model. One example of graphical model is proposed by `ContextUML` [SB05] where models are used to separate the definition and information related to the context from the specific implementation. Another example that uses special diagram elements and relationships is found in [HIR02]. Here it is remarked that a context model should also differentiate between static and dynamic information. Our metamodel, presented in section 3, is an extended version of the metamodel introduced in [PDNP07]. The main purpose of our metamodel is to provide a language to build models with context information and use them as inputs to model to model and model to code transformations. We separate particular concepts important in our adaptation approach like *global* and *local* information or *private* and *shared* user data. These concepts are evaluated when transforming the models and as a result of such evaluation, generated components change in correspondence with context information.

5 Conclusions and Perspectives

In this proposal paper, we have discussed several challenges on ubiquitous applications and pervasive environments. We have discussed about adaptability of software as an appropriate solution to deal with mobility. Our proposal to achieve adaptation is based on the generation and deployment of an initial bootstrap, which enforces common understanding of context information. A metamodel for context information has been described to separate context information different categories. The main motivation behind our approach is to deal with open environments in which services and resources available vary frequently in response to user movements or changes in the surrounding context. In such a scenario, it is hard for devices to take advantage of the services. Nevertheless, by loading an extendable bootstrap application, it is imaginable to create a process of adaptation in which every mobile device learns how to manage context information so it can communicate with different providers and get access to customized information and services.

Acknowledgements: This work is part of the `CAPPUCINO` project, funded by the *Conseil Régional Nord-Pas-de-Calais*, *Oseo/ANVAR*, and the *Fonds de Compétitivité des entreprises*

Bibliography

[Cap08] Cappucino. The Cappucino Project. http://cappucino.oqube.com/, 2008.

[FGM05] A. Flissi, C. Gransart, P. Merle. A Component-based Software Infrastructure for Ubiquitous Computing. In *4th International Symposium on Parallel and Distributed Computing (ISPDC 2005)*. Lille, France, jul 2005.

[HIR02] K. Henricksen, J. Indulska, A. Rakotonirainy. Modeling context information in pervasive computing systems. In Springer (ed.), *1st International Conference on Pervasive Computing*. Zurich, Switzerland, 2002.

[MM03] J. Miller, J. Mukerji. MDA Guide Version 1.0.1. Technical report, Object Management Group (OMG), 2003.

[Pas08] L. Passani. WURFL. http://wurfl.sourceforge.net/index.php, 2008.

[PDNP07] C. A. Parra, M. D'Hondt, C. Noguera, E. V. Paesschen. Introducing Context-Awareness in Applications by Transforming High-Level Rules. In *Object Technology for Ambient Intelligence Workshop (OT4AmI)*. Berlin, Germany, 2007.

[PSKC04] Philip, S. M. Sadjadi, E. P. Kasten, B. H. C. Cheng. A Taxonomy of Compositional Adaptation. Technical report MSU-CSE-04-17, Michigan State University, Department of Computer Science and Engineering, East Lansing, Michigan, 2004.

[SB05] Q. Z. Sheng, B. Benatallah. ContextUML: a UML-based modeling language for model-driven development of context-aware Web services. *Mobile Business, 2005. ICMB 2005. International Conference on*, 2005.

[SP04] T. Strang, C. L. Popien. A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*. Pp. 34–41. Nottingham, England, September 2004.

[Szy98] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Professional, December 1998.