MobiCASE 2010
Workshop on Mobile Software Engineering
Position Paper

Paper title: **MDE, DSL and Tooling for Effective Context Management in Ubiquitous Computing**

Name: Sophie Chabridon, Denis Conan, Chantal Taconet, Cong Kinh Nguyen, Cao Cuong Ngo, Léon Lim, and Zied Abid
Role/Title: Academic Research Team
Organization: *Institut Telecom / Telecom SudParis / SAMOVAR Lab*
Preferred Contact: Sophie.Chabridon@institut-telecom.fr

1. Your experience as a software engineer?

- Open-source framework COSMOS (COntext entitieS coMpositiOn and Sharing) [1,2,3]:

COSMOS is component-based framework for managing context data in ubiquitous environments through the concepts of context node and context management policies translated into software components and software architectures. It enforces recursion and component sharing. Recursion allows components to be nested within composite components. With sharing, a given component can be included (or shared) by several parent (or composite) components. In addition, we favor flexible architectural patterns and try to save system resources in order to be able to reach a good level of scalability without degrading performances when the number of observed context sources and context processors becomes very large. Moreover, COSMOS reorganizes the classical functionalities of a context manager to systematically introduce a 3-steps cycle of data collection, data interpretation, and situation identification. Although situation identification actions should not be too frequent, processing context information is an activity that must be conducted more often, while data gathering is a third activity that must be continuous. Thus, we have three different types of activities with different frequencies. We decouple as much as possible these activities in order to obtain a non-blocking and configurable framework.

- MDE with a DSL [3,4]:

For the sake of usability, a context management framework must provide to designers and developers an easy way to specify the context entities they want to observe and the context data they want to compute. In our approach, designers and developers specify their monitoring requirements in contracts that we call "observation contracts".

Nevertheless, in very dynamic environments encountered in ubiquitous computing, the concept of observation contract is complex to define due the volatility of context data sources, the multiplicity of context data sources that may provide somewhat incoherent context data, the conflicts appearing between application requirements, etc. This very high dynamicity requires that the context management service must adapt itself to context changes: new context data sources, new clients, new observation contracts, new network topology changes, etc.

In addition, this complexity needs modeling and analysis. Both previous issues fit well to autonomous computing principles [5]: models of observation contracts are built at design time and reified at run-time to allow run-time analysis and planning of adaptations to the software architecture of the context management service. Moreover, for better productivity, designers and developers write their observation contracts in a domain specific language, COSMOS DSL.

- Tooling

The COSMOS ecosystem benefits from many tools provided by the open source community. Software project management is taken into account by Apache Maven [6]. COSMOS DSL is specified in Eclipse Xtext [7] as an ANT-LR grammar annotated with meta-model elements described using the Eclipse Modeling Framework (EMF) [8]. We have written several Maven plug-ins to generate COSMOS artifacts (skeletons of primitive components and definition of the software architecture of composite components), and to transform Java 1.5 code into code compatible with the J2ME CLDC profile. We also use many existing plug-ins for instance to package COSMOS applications for MID-Let phones and Android phones. In addition, in order to exercise our context management policies by simulation before deploying them in physical environments, we use the Siafu context simulator [9] for writing scenarios.

2. Your experience with mobile software development?

Our developments target desktops, laptops, Internet tablets and mobile phones. The corresponding operating systems are Windows, GNU/Linux, Symbian OS, Windows Mobile, and Android. The application domains where COSMOS is applied by academic partners and by industrial partners are mobile commerce, mobile learning and pervasive gaming.

3. How does traditional software engineering relate to the engineering of mobile applications and systems?

In the software engineering approach chosen for the engineering of our context sensitive applications , we promote many software engineering concepts, disciplines and technologies: component-based software engineering, architectural patterns, design patterns, idioms, model driven engineering, domain specific languages, control loop of autonomous computing, etc. A key issue is to relate these concepts, disciplines and technologies to others in usage in the communities of mobile phones and small devices programmers: for instance, rapid prototyping and scripting to name a few. Our main role is to do research for asserting a given level of quality of mobile applications developed, deployed and used.

4. What are the distinguishing features of mobile software specification, architecture, development and testing that need special attention, skills, or innovation?

Rapid prototyping in addition to simulation and testing is compulsory to get users' feedback as soon as possible on real mobile devices since emulation environments are not sufficient. To assert a given level of quality while using rapid prototyping approaches, we think that the issues listed in the next section must be addressed.

5. What is the suggested focus and agenda for mobile software engineering research and education?

At the workshop, we would like to discuss the following issues that we think are important:

- MDE, DSL and extensions of IDE environments to reach mature graphical editors for CBSE.
- Complex context simulation taking into account user and system environments.
- Specification of autonomous computing capabilities by non-programmers.

References :

[1] D. Conan, R. Rouvoy, and L. Seinturier. Scalable Processing of Context Information with COSMOS. In Proc. 6th IFIP WG 6.1 International Conference DAIS, vol. 4531 of LNCS, pages 210–224, June 2007.

[2] COSMOS framework http://picoforge.int-evry.fr/projects/svn/cosmos

[3] R. Rouvoy, D. Conan, and L. Seinturier. Software Architecture Patterns for a Context Processing Middleware Framework. IEEE Distributed Systems Online, 9(6), June 2008.

[4] L. Lim, Description of context management policies (in French), Master's thesis, Université d'Évry-Val d'Essonne, Télécom SudParis, Évry, France, July 2008.

[5] J.O. Kephart and D.M. Chess, The Vision of Autonomic Computing, IEEE Computer, 36(1), January 2003, pages 41-50.

[6] Apache Maven http://maven.apache.org

[7] Xtext - Language Development Framework http://www.eclipse.org/Xtext

[8] Eclipse Modeling Framework Project (EMF) http://www.eclipse.org/modeling/emf

[9] Siafu context simulator http://siafusimulator.sourceforge.net