Chapter 29
Designing Mobile Aspect-Oriented Software Architectures with Ambients

Nour Ali
Lero - The Irish Software Engineering Research Centre, University of Limerick, Ireland

Isidro Ramos
Polytechnic University of Valencia, Spain

ABSTRACT

This chapter focuses on designing software architectures of mobile applications using an Aspect-Oriented Architecture Description Language (AOADL). The AOADL follows an approach called Ambient-PRISMA which enables designers to address, in an explicit and abstract way, the notion of location and mobility. Concretely, the AOADL extends the PRISMA AOADL by introducing a primitive called an ambient which is inspired by Ambient Calculus. An ambient defines a bounded place where other architectural elements (components and connectors) reside and are coordinated with elements that are outside an ambient's boundary. Architectural elements can enter and exit ambients. Ambients, as well as other architectural elements, are defined by importing aspects. Thus, behaviours that change the location of architectural elements are specified separately in distribution aspects. The objective of this chapter is to explain the steps that have to be followed when designing architecture configurations of distributed and mobile systems using the Ambient-PRISMA AOADL. This is explained by using a running example of a distributed auction system.

1. INTRODUCTION

In the last few decades, the information society has undergone important changes that have increased the complexity of software development. The software, the devices (PCs, laptops, PDAs, smart phones, etc) and people involved in current business processes are often distributed, and mobile. As a result, the structure of software systems has become complex due to the new requirements that these systems need to satisfy such as mobility and adaptability.

Software architecture is a discipline that focuses on the design and specification of overall

DOI: 10.4018/978-1-61520-655-1.ch029

system structure (Shaw & Garlan, 1996) (Bass, Clements, & Kazzman, 2003). It is considered to be the bridge between the requirements and implementation phases of the software life cycle. The software architecture of a system describes its structure in terms of components (computational units), connectors (coordination units), and configurations (connections of components and connectors) (N. Medvidovic & Taylor, 2000).

Mobility causes changes in the structure of a distributed software system (Carzaniga, Picco, & Vigna, 1997). In this way, software architecture techniques can be useful to support the representation of these changes and the design of mobile software systems. Software architecture approaches have been proposed for modeling mobility (e.g. MobiS (Ciancarini & Mascolo, 1998) and Community (A. Lopes, Fiadeiro, & Wermelinger, 2002)). A comparison between different approaches has been made in (Ali, Solis, & Ramos, 2008). It can be inferred from this comparison that most of the proposed approaches do not provide an explicit notion for the location and mobility primitives.

An approach which provides an explicit notion of location and mobility is Ambient Calculus (AC) (Cardelli, 1999; Cardelli & Gordon, 1998), which is a process algebra which provides a primitive called ambient that represents a bounded place where computation happens. Ambients can form hierarchies and mobility is modelled by using ambient capabilities. The exit capability allows an ambient to exit its parent ambient. The enter capability provides an ambient to move in a sibling ambient in the ambient hierarchy.

A technique that aims to reduce complexity by increasing reusability, flexibility, and maintainability through the software development process is Aspect-Oriented Software Development (AOSD) (Filman, Elrad, Clarke, & Aksit, 2004). AOSD allows the separation of concerns by modularizing crosscutting concerns in separate entities called aspects (Kiczales, et al., 2001). Distribution and mobility have been identified as crosscutting concerns (Lobato, et al., 2004; C. V. Lopes, 1997; Soares & Borba, 2002); separating them from other concerns of the software system will increase their reusability, and decrease cost and effort to maintain them.

Some researchers have proposed the integration of AOSD and software architecture (Chitchyan, 2005; Cuesta, del Pilar Romay, de la Fuente, & Barrio-Solórzano, 2005). PRISMA is an approach that integrates AOSD and Component Based Software Development (CBSD) for describing software architectures (Pérez, et al., 2008). The crosscutting concerns of the software architecture are specified in aspects. Architectural elements (components and connectors) are then defined by using aspects.

Ambient-PRISMA (Ali, 2008) enriches PRISMA with the ambient concept inspired from Ambient Calculus (AC) in order to design, in an explicit and abstract way, the notion of location and mobility of architectural elements. AMBIENT-PRISMA introduces ambients as new kinds of architectural elements which define a bounded place where other architectural elements reside and can be coordinated with the exterior. Ambients can be hierarchically organized, conforming to a tree structure which is used to model distributed systems hierarchies. Architectural elements (including other ambients) can move by entering and exiting ambients. The functionality (behaviour) of an ambient is defined through mobility, coordination and distribution aspects.

This chapter focuses on explaining the steps that have to be followed by a user of the Ambient-PRISMA approach to specify mobility properties and design topologies of software architectures of mobile systems. The AOADL primitives will be illustrated by designing the aspect-oriented architecture of a distributed auction system with mobile agents.

The structure of the chapter is the following: Section 2 gives an overview of Ambient-PRISMA. Section 3 presents the distributed auction system