Chapter 92
Reusing Services through Context-Aware Discovery and Adaptation in Pervasive Systems

Javier Cubo
University of Málaga, Spain

Ernesto Pimentel
University of Málaga, Spain

ABSTRACT

Reusing of software entities, such as components or services, to develop software systems has matured in recent years. However, it has not become standard practice yet, since using pre-existing software requires the selection, composition, adaptation, and evolution of prefabricated software parts. Recent research approaches have independently tackled the discovery, composition, or adaptation processes. On the one hand, the discovery process aims at discovering the most suitable services for a request. On the other hand, the adaptation process solves, as automatically as possible, mismatch cases which may be given at the different interoperability levels among interfaces by generating a mediating adaptor based on an adaptation contract. In this chapter, the authors present the DAMASCo framework, which focuses on composing services in mobile and pervasive systems accessed through their public interfaces, by means of context-aware discovery and adaptation. DAMASCo has been implemented and evaluated on several examples.

INTRODUCTION

The increased usage of mobile and portable devices has given rise over the last few years to a new market of mobile and pervasive applications (Wang et al., 2007). These applications may be executed on either mobile computers (laptops, notebooks, tablet PCs, etc.), or wireless hand-held devices (PDAs -Personal Digital Assistants-, smart phones, etc.), or embedded systems (on-board computer, household appliances, intelligent buildings, etc.), or even on sensors or RFID tags. Their main goal is to provide connectivity and services at all time, adapting and monitoring when required and improving the user experience. These systems are different to traditional distributed computing

DOI: 10.4018/978-1-4666-4301-7.ch092

systems. On the one hand, a mobile system is able to change location allowing the communication via mobile devices. This results in some new problems which need to be resolved (connectivity, bandwidth, local resources, etc.). On the other hand, a pervasive application attempts to create an ambient intelligence environment to make the computing part of it and its enabling technologies essentially transparent.

Context-aware computing (Dargie, 2009) covers all the topics related to the building of systems which are sensitive to their context (location, identity, time and activity), by adapting their behaviour at run-time according to the changing conditions of the environment, as well as those of user preferences or privileges (Schilit el al., 1994). Such computing is incorporated into mobile and pervasive systems, with the inclusion of certain context-awareness features. One of the main features of context information is its dynamism. Thus, context-aware applications should support a high-level of adaptation in the values that define the context information. Adaptations may be as varied as the information related to the context. For example, a context-aware mobile device which detects that the user is in a meeting might vibrate instead of sounding. An application running on a PDA could adjust the level of screen brightness according to ambient light. An application that requires a server could make a cache of data to address the connection loss related to the wireless networks.

Developing real-world mobile and pervasive systems taking into account all the aforementioned concerns is extremely complex and error-prone. Therefore, it is essential to determine an effective methodology to develop software. In order to reduce efforts and costs, context-aware systems may be developed using existing Commercial-Off-The-Shelf (COTS) components or (Web) services, since reusability is considered one of the main concerns in software engineering that determines the quality of software. In contrast to the traditional approach in which software systems are implemented from scratch, COTS and services can be developed by different vendors using different languages and different computer platforms. Although the reuse of software has matured and has overcome the previously mentioned problems, it has not become standard practice yet, since reusing components or services requires the selection, composition, adaptation and evolution of prefabricated software parts. Component-Based Software Engineering (CBSE) (Szyperski, 2003), also known as Component-Based Software Development (CBSD), and Service-Oriented Architecture (SOA) (Erl, 2005) promote software reuse by selecting and assembling pre-existing software entities (COTS and services, respectively). Thus, these software development paradigms allow building fully working systems as efficient as possible in order to improve the level of software reusability.

While composing services, issues related to faults or changes arise dynamically and continuously, and they have to be detected and handled. These issues can be:

- Mismatch problems,
- Requirement and configuration changes,
- Network and remote system failures, and
- Internal service errors.

In this chapter, we focus mainly on avoiding mismatches. A proposal as an initial attempt to solve the third and fourth type of problems can be found in (Cubo et al., 2010; Cubo et al., 2011). Unfortunately, in most cases it is impossible to modify services in order to adapt them, since their internal implementation cannot be inspected or modified. Thus, due to the black-box nature of the services that compose the mobile and pervasive systems, they must be equipped with external interfaces giving information about service functionality. In particular, the interfaces of the constituent services of a system do not always fit one another and some features of these services may change at run-time, therefore they require a certain degree of adaptation in order to avoid