Chapter 6
A Software Engineering Framework for Context-Aware Service-Based Processes in Pervasive Environments

Zakwan Jaroucheh
Edinburgh Napier University, UK

Xiaodong Liu
Edinburgh Napier University, UK

Sally Smith
Edinburgh Napier University, UK

ABSTRACT

Context-awareness is considered to be the cornerstone technique for developing pervasive computing applications that are flexible, adaptable, and capable of acting autonomously on behalf of the user. However, context-awareness introduces various software engineering challenges. The separation of concerns is a promising approach in the design of the context-aware adaptive processes (CAAPs) where the core logic is designed and implemented separately from the context handling and adaptation logics. In this respect, this chapter presents a conceptual framework for developing CAAPs and software infrastructure for efficient context management that together address the known software engineering challenges and facilitate the design and implementation tasks associated with such context-aware applications.

INTRODUCTION

Context-awareness refers to the capability of an application or a service being aware of its physical environment or situation (e.g. context) and to respond proactively and intelligently based on this awareness (Baldauf et al. 2007). It is widely acknowledged that, compared to desktop applications, pervasive environments introduce a new wave of software engineering challenges.

Firstly, in such highly dynamic environments the ultimate objective is to amplify human activi-

DOI: 10.4018/978-1-60960-735-7.ch006

ties and demanding minimal attention from the user. Context-aware applications aims to meet these objectives or requirements by adapting to a subset of the current context considered relevant to the task at hand such as the user location, time, and user activity. In this chapter we focus on developing context-aware service-based processes applications. In addition, we define the context-aware process adaptation as *the action that modifies the process in a way that causes process behavior to evolve according to the evolution of business and users' requirements, and the context considered relevant to that process*. To this end, process modeling must be flexible enough to deal with constant changes – both at the business level (e.g. evolving business rules) and the technical level (e.g. contextual information and platform upgrades). The flexibility could be provided or addressed by incorporating variabilities into a system (Koning et al. 2009). Therefore, the notion of an *evolution fragment* and *evolution primitive* that capture the process variability in a logical way are introduced.

Secondly, in order to have the transition of the context-aware applications out of the laboratory to the marketplace, there is a need for a software engineering framework that simplifies the design and implementation of context-aware software. To this end, the approach proposed in this chapter could apply an adaptation to processes modeled or developed without any adaptation possibility in mind and independently of specific usage contexts. In addition, it supports the viewpoint of context-aware adaptation as a crosscutting concern with respect to the core "business logic" of the process. In this way, the design of the process core can be decoupled from the design of the adaptation logic which significantly eases the process design and rapid prototyping.

Thirdly, to ease the development of such applications it is necessary to decouple the application from context acquisition and representation, and at the same time provide universal models and mechanisms to manage context. Thus, generic and dynamically manageable context models are of interest since they can be reused by different applications and ease context sharing between systems (Chen et al. 2004). Therefore, in this chapter, we present a context management infrastructure based on a flexible product line based context model which significantly enhances reusability of context information by providing context variability constructs to satisfy different application needs.

The rest of the chapter is organized as follows: In "Context Management Framework," the context management infrastructure is described. It this section the conceptual model for context management is introduced as well as the rationale behind it; then we describe how to model the context information considering the application requirement perspective. For this purpose we leverage ideas from software product line techniques. In "A Model-Driven Framework for Managing Context-Aware Adaptive Service-Based Processes," a MDD-based (Model-Driven Development) framework called *Apto* (Latin word for Adapt) is introduced. We show how to capture the variability in the service-based process in a logical form by introducing the notion of context fragments and context primitives. The proposed approach contributes to a solution to automatically generating a customized process based on the context. Another feature is that Apto supplies a set of automated tools for generating and deploying executable process definitions e.g. WS-BPEL (OASIS, 2007) which in turn significantly reduces the development cost. In "Case Study," we illustrate the proposed frameworks by giving a simple example of Conference Event Advisor process. The related work and concluding remarks ends the chapter.