Chapter 6
Rapid Development of Adaptable Situation-Aware Service-Based Systems

S. S. Yau  
Arizona State University, USA

S. Mukhopadhyay  
Louisiana State University, USA

H. Davulcu  
Arizona State University, USA

D. Huang  
Arizona State University, USA

R. Bharadwaj  
Naval Research Laboratory, USA

K. Shenai  
University of Toledo, USA

ABSTRACT

Service-based systems have many applications, such as collaborative research and development, e-business, health care, military applications and homeland security. In these systems, it is necessary to provide users the capability of composing appropriate services into workflows offering higher-level functionality based on declaratively specified goals. In a large-scale and dynamic service-oriented computing environment, it is desirable that the service composition is automated and situation-aware so that robust and adaptive workflows can be generated. However, existing languages for web services are not expressive enough to model services with situation awareness (SAW) and side effects. This chapter presents an approach to rapid development of adaptable situation-aware service-based systems. This approach is based on the $\alpha$-logic and $\alpha$-calculus, and a declarative model for SAW. This approach consists of four major components: (1) analyzing SAW requirements using our declarative model for SAW, (2) translating the model representation to $\alpha$-logic specifications and specifying a control flow graph

DOI: 10.4018/978-1-61520-684-1.ch006
Rapid Development of Service-Based Systems

**INTRODUCTION**

Nowadays, heterogeneous networked systems involving sensors, actuators, access points, and control stations connected through wired/wireless networks are being widely used in many mission-critical applications, such as healthcare, military, automobile and space applications. These systems often need to satisfy various requirements simultaneously, such as real-time, fault-tolerance, and security, and may need to be run in resource-poor environments with bandwidth limitations and shortage of memory space and/or other critical resources. Such systems often need to monitor physical and/or computational contexts, which are any instantaneous, detectable, and relevant properties of the environment, the system, or the users (Yau, Karim, Y. Wang, B. Wang, & Gupta, 2002; Yau, Wang & Karim, 2002), and react to situation changes in an autonomous and distributed manner. A situation is a set of contexts in a system over a period of time that affects future system behavior for specific applications (Yau, Karim, Y. Wang, B. Wang, & Gupta, 2002; Yau, Wang & Karim, 2002). Components of these systems can migrate from one physical/logical location to another in runtime. Hence, it is of utmost importance that such systems are adaptable and situation-aware so that they can provide services to users reliably and intelligently in accordance with the user goals in harsh and rapidly-changing environments.

One of the most promising architectures for developing such adaptable situation-aware systems is the Service-Oriented Architecture (SOA) (W3C, 2004b) due to the major advantage of enabling rapid composition of distributed applications from available services, regardless of the programming languages and platforms used in developing and running the applications. The systems based on SOA are called service-based systems (SBS). In SBS, various capabilities are provided by different organizations as services, which are software/hardware entities with well-defined interfaces to provide certain functionality. These services can be used to compose higher-level functionality representing a certain workflow, which is a series of cooperating and coordinated activities achieving certain user goals. Hence, the development of an SBS can be viewed as a process involving the specification of user goals, the identification of suitable services, the composition of identified services into workflows satisfying the specified goals, and the implementation, testing and validation of the service composition.

A workflow can often be specified by a control flow graph (Hsu, 1993; Georgakopoulos, Hornick, & Sheth, 1995; WFMC, 1996), which is a common type of framework used in most workflow management systems for depicting the local execution dependencies among key activities in the workflow. A typical control flow graph specifies the initial and final activities in a workflow, the successor-activities for each activity in the graph, and branch selection or parallel execution of the successor-activities. However, dependencies among service invocations in a workflow in adaptable situation-aware SBS are often based on situation changes in highly dynamic environments, where users often have different requirements in various situations, or services cannot provide desirable QoS due to attacks, system failures or overload. Such dependencies can be effectively...