Automated Situation-Aware Service Composition in Service-Oriented Computing

S. S. Yau, Arizona State University, USA
H. Davulcu, Arizona State University, USA
S. Mukhopadhyay, Utah State University, USA
D. Huang, Arizona State University, USA
H. Gong, Arizona State University, USA
P. Singh, Arizona State University, USA
F. Gelgi, Arizona State University, USA

ABSTRACT

Service-based systems have many applications, such as e-business, health care, and homeland security. In these systems, it is necessary to provide users the capability of composing services into workflows providing higher-level functionality. In dynamic service-oriented computing environments, it is desirable that service composition is automated and situation-aware to generate robust and adaptive workflows. In this paper, an automated situation-aware service composition approach is presented. This approach is based on the $\alpha$-logic, $\alpha$-calculus, and a declarative model for situation awareness (SAW). This approach consists of four major components: (1) analyzing SAW requirements using our SAW model, (2) translating our SAW model representation to $\alpha$-logic specifications and specifying a control flow graph in $\alpha$-logic as the service composition goal, (3) automated synthesis of $\alpha$-calculus terms defining situation-aware workflow agents based on $\alpha$-logic specifications for SAW requirements and the control flow graph, and (4) compilation of $\alpha$-calculus terms to executable components.

Keywords: $\alpha$-calculus; $\alpha$-logic; agent synthesis; automated situation-aware service composition; service-based systems; situation-aware workflow

INTRODUCTION

Service-oriented architecture (SOA) (W3C, 2004b) has the major advantage of enabling rapid composition of distributed applications, regardless of the programming languages and platforms used in developing and running the applications. SOA has been adopted in many distributed systems like Grid and Global Information Grid (GIG) (U.S. Department of Defense, 2002), in various application domains, such as collaborative research and development, e-business, health care, environmental control,
military applications, and homeland security. The systems based on SOA are considered as service-based systems (SBS). In SBS, capabilities are provided by various organizations as services, which are software/hardware entities with well-defined interfaces to provide certain capability over wired or wireless networks using standard protocols, such as HTTP and Simple Object Access Protocol (SOAP). For a user to effectively utilize available services in an SBS to achieve the user’s goal, it is necessary to provide the capability for the user to compose appropriate services into higher-level functionality. Such higher-level functionality is considered as a workflow, which is a series of cooperating and coordinated activities. Since a large-scale SBS usually consists of thousands of services, it is desirable that the SBS can allow users to declaratively specify their goals and automate the service composition based on the specified goals.

Control flow graph (Georgakopoulos et al., 1995; Hsu, 1993; WFMC, 1996) is a common type of framework for depicting the local execution dependencies among the key activities in a service composition. It is a good way to visualize the overall flow of control among the milestones that a workflow should satisfy. Control flow graphs are the primary specification means in most commercial implementations of workflow management systems. A typical graph specifies the initial and final activities in a workflow, the successor-activities for each activity in the graph, and the execution of all the successors concurrently or only a selected successor.

On the other hand, dependencies among service invocations in a workflow 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 captured by situational constraints, which are the restrictions on what and how services should be invoked in various situations. A situation is a set of contexts in a system over a period of time that affects future system behavior for specific applications, and a context is any instantaneous, detectable, and relevant property of the environment, the system, or the users (Yau, Wang, & Karim, 2002; Yau et al., 2002). It is necessary to specify situational constraints in a modular and reusable fashion so that a service composition system can discover appropriate services based on situations and users’ goals, compose services into a control flow, and coordinate their execution adaptively based on situation changes. We consider such a process as situation-aware service composition.

So far, the most widely used specification language for Web services, WSDL (WSDL, 2001), is not expressive enough to specify these situational constraints. Other frameworks for modeling and executing workflows in Web service based systems, such as BPEL4WS (Andrews et al., 2003) and OWL-S (W3C, 2004a), are not expressive enough to model services with side effects (i.e., services can change states of themselves or other services), and do not provide any facilities for automated service composition.

In this article, we will present an approach to automated situation-aware service composition which is based on our $\alpha$-logic, $\alpha$-calculus, and declarative model for situation awareness (SAW) (Yau et al., 2005a; Yau et al., 2005b). Our approach will include an efficient and constructive agent synthesis algorithm based on $\alpha$-logic proof theory that can automate the synthesis of executable agents which satisfy the control flow and situational constraint specifications for situation-aware service composition.

In the following sections, we will summarize the key concepts and features of our SAW model, $\alpha$-logic and calculus. We will show how to use our SAW model and $\alpha$-logic to specify situational constraints and control flow graphs. We will present an algorithm for automated situation-aware service composition based on $\alpha$-logic proof theory. We will also show how situation-aware workflow agents described in $\alpha$-calculus terms are synthesized from the workflows generated using our service composition algorithm. These agents will be used to monitor and execute the workflows.
SOA Reference Architecture

www.igi-global.com/chapter/soa-reference-architecture/41515?camid=4v1a