Information Sciences 258 (2014) 416–433

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Behavior modeling and automated verification of Web services

Quan Z. Sheng [a,*], Zakaria Maamar [b], Lina Yao [a], Claudia Szabo [a], Scott Bourne [a]

[a] School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
[b] College of Information Technology, Zayed University, Dubai, United Arab Emirates

ARTICLE INFO

Article history:
Available online 27 September 2012

Keywords:
Web service
Cloud computing
Service behavior
Conversation message
Symbolic model checking

ABSTRACT

Cloud computing has been rapidly adopted over the last few years. However, techniques on Web services, one of the most important enabling technologies for cloud computing, are still not mature yet. In this paper, we propose a novel approach that supports dependable development of Web services. Our approach includes a new Web service model that separates service behaviors into operational and control behaviors. The coordination of operational and control behaviors at runtime is facilitated by conversational messages. We also propose an automated service verification approach based on symbolic model checking. In particular, our approach extracts the checking properties, in the form of temporal logic formulas, from control behaviors, and automatically verifies the properties in operational behaviors using the NuSMV model checker. The approach presented in this paper has been implemented using a number of state-of-the-art technologies. We conducted a number of experiments to study the performance of our proposed approach in detecting design problems in services. The results show that our automated approach can successfully detect service design problems. Our system offers a set of tools assisting service developers in specifying, debugging, and monitoring service behaviors.

1. Introduction

Over the past few years, cloud computing is gaining a considerable momentum as a new computing paradigm for providing flexible and dynamic services and infrastructures on demand [2,33,41]. Cloud computing holds the potential to transform the landscape of the IT industry by making software more attractive as services and shaping the way IT hardware is designed and purchased.

Service-oriented architecture (SOA) and Web services in general are one of the most important enabling technologies for cloud computing in the sense that resources (e.g., software, infrastructures, and platforms) are exposed in the clouds as services [37]. Most research in cloud computing so far focuses on topics such as virtulization, reliability, scalability, security and privacy of cloud services [28,25,20,30]. Although these are all important, Web services, the fundamental topic that underpins the cloud computing paradigm, have not received enough attention. In fact, despite active research into, and development of, Web services over the last decade, Web services are still not fully mature yet. According to a recent study in Europe [11], the Web currently contains 30 billion Web pages, with 10 million new pages added each day. In contrast, only 12,000 Web services exist on the Web. Even worse, most of these Web services have been deployed with dependability problems (e.g., unexpected behaviors, delayed or even no responses) [36,27,40]. One significant challenge is that, to the best of our knowledge,

* Corresponding author.
E-mail addresses: qsheng@cs.adelaide.edu.au (Q.Z. Sheng), zakaria.maamar@zu.ac.ae (Z. Maamar), lina.yao@adelaide.edu.au (L. Yao), claudia.szabo@adelaide.edu.au (C. Szabo), scott.bourne@adelaide.edu.au (S. Bourne).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.09.016

there lacks of novel approaches and tools that would enable service developers to check the soundness and completeness of their services design so that the design problems can be identified and addressed at early stages, which ultimately ensuring the quality of the services released to clouds. Given the quick adoption of cloud computing in industry (e.g., Amazon Web Services, Google AppEngine, and Microsoft Azure), more and more cloud services will emerge, which support the development of numerous applications including mission-critical applications such as health care, air traffic control, and stock trading. This calls for the urgent need to develop novel techniques for producing highly dependable cloud services.

In this paper, we present our approach on modeling Web services so that design problems and errors can be early identified and addressed. In particular, we propose to divide Web service behaviors into two types: *operational behaviors* and *control behaviors*, based on the separation of concerns design principle [17]. The operational behavior, which is application dependent, illustrates the business logic that underpins the functioning of a Web service. The control behavior, which is application independent, acts as a controller over the operational behavior and guides its execution progress. The interactions between control and operational behaviors are modeled as *conversation sessions* (i.e., sequences of messages exchanged between the control and operational behaviors). By analyzing conversational messages and checking service behavior specifications, it is possible to verify the service design. The main contributions of this paper are as follows:

- A service behavior model that decouples operational and control behaviors of Web services. This separation of Web service behaviors eases not only the development and maintenance, but the verification (e.g., soundness and completeness checking), testing, and debugging of Web services. To the best of our knowledge, this is the first effort that identifies two behaviors of Web services.
- A service verification approach based on symbolic model checking [10]. Our approach extracts the checking properties, in the form of temporal logic formulas, from control behaviors, and automatically verifies the properties in operational behaviors.
- A fully functional prototype system that offers a set of tools for the specification of Web services and automated verification of the service design.

The reminder of the paper is organized as follows. Section 2 describes the details of our new Web service behavior model. Section 3 presents a symbolic model checking approach for verifying service designs. Section 4 focuses on the implementation and validation of the proposed system. Finally, Section 5 overviews related work and Section 6 provides some concluding remarks.

2. Service behavior model

Cloud services are normally exposed as Web services that follow the industry standards such as Web Services Description Language (WSDL). Unfortunately, WSDL does not show how they function or how their executions can be overseen. As a result, Web services are still largely perceived as simple, passive components that react upon request only [8,40]. In this section, we present a Web service behavior model using more richer description, which isolates a service from any orchestration scenario before it abstracts and separates its behavior into operational behavior and control behavior. We use statecharts [14] to model both behaviors. It should be noted that other formalisms such as Petri nets [21] also can be used. A magazine version of the content in this section appears in [31]. In this paper, we present a complete and more formal description of the service behavior model.

2.1. Operational and control behaviors

The *operational behavior* shows the business logic that underpins the functioning of a Web service. In contrast, the *control behavior* guides the execution progress of the business logic of a Web service. The control behavior relies on a number of states (`activated`, `not-activated`, `done`, `aborted`, `suspended`, and `compensated`) that are reported in the transactional Web services literature [22,23,38].

Definition 1 (*Web Service Behavior*). The behavior of a Web service is a 5-tuple $\mathcal{B} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s^0, \mathcal{F} \rangle$ where:

- $\mathcal{S}$ is a finite set of state names.
- $s^0 \in \mathcal{S}$ is the initial state.
- $\mathcal{F} \subseteq \mathcal{S}$ is a set of final states.
- $\mathcal{L}$ is a set of transition labels.
- $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is the transition relation. Each transition $t = (s^{src}, l, s^{tgt})$ is composed of a source state $s^{src} \in \mathcal{S}$, a target state $s^{tgt} \in \mathcal{S}$, and a transition label $l \in \mathcal{L}$. We qualify these transitions as *intra-behavior* (the rationale behind this qualification will be given later).

In statecharts, a label consists of three optional components (event E, condition C, and action A) and is written as $E[C]/A$.

A Web service's control and operational behaviors are instances of the Web service's behavior. These two behaviors are denoted by $\mathcal{B}_{co} = \langle \mathcal{S}_{co}, \mathcal{L}_{co}, \mathcal{T}_{co}, s^0_{co}, \mathcal{F}_{co} \rangle$ and $\mathcal{B}_{op} = \langle \mathcal{S}_{op}, \mathcal{L}_{op}, \mathcal{T}_{op}, s^0_{op}, \mathcal{F}_{op} \rangle$, respectively.