Title: Broker-based SLA-aware composite service provisioning

Author: Quanwang Wu Qingsheng Zhu Xing Jian Fuyuki Ishikawa

PII: S0164-1212(14)00145-9
DOI: http://dx.doi.org/doi:10.1016/j.jss.2014.06.027
Reference: JSS 9346

To appear in:

Received date: 1-8-2013
Revised date: 6-6-2014
Accepted date: 16-6-2014

Please cite this article as: Wu, Q., Zhu, Q., Jian, X., Ishikawa, F.,Broker-based SLA-aware composite service provisioning, *The Journal of Systems and Software* (2014), http://dx.doi.org/10.1016/j.jss.2014.06.027

Broker-based SLA-aware composite service provisioning

Quanwang Wu[1,2], Qingsheng Zhu[1], Xing Jian[1], Fuyuki Ishikawa[2]

[1] *Chongqing Key Laboratory of Software Theory & Technology, Chongqing University, Chongqing, China*

[2] *National Institute of Informatics, Tokyo, Japan*

Abstract

QoS-aware service composition aims to satisfy users' quality of services (QoS) needs during service composition. Traditional methods simply attempt to maximize user satisfaction by provisioning the composite service instance with the best QoS. These "best-effort" methods fail to take into account that there also exist other consumers competing for the service resources and their decisions of service selection / composition can impact on QoS. Since user's QoS needs can be met once the demanded level is reached, in this paper, we propose an "on-demand" strategy for QoS-aware service composition to replace the traditional "best-effort" strategy. The service broker is introduced to facilitate implementation of this strategy: it first purchases a number of service instances for each component from providers and then provisions the composite services with different QoS classes to consumers. This paper focuses on how the broker follows the service level agreement (SLA) to provision composite services in the "on-demand" manner. This problem is formally expressed as the minimization of the QoS distance function between SLA and QoS of composite service instances, under a series of constraints. Heuristic approaches are proposed for the problem and experiments are conducted at last to verify their effectiveness and efficiency.

Keywords: composite service, broker, SLA, QoS, on-demand

1. Introduction

The Service-Oriented Computing (SOC) paradigm utilizes services as fundamental elements to support rapid and low-cost development of distributed applications in heterogeneous environments (Papazoglou et al., 2007). As one of the key concepts in SOC, service composition aims to orchestrate multiple elementary services to fulfill a sophisticated application. The functional template of composite services can be first defined as a set of interconnected abstract services, and at running time, each abstract service is dynamically bound to a concrete service so that the instantiated composite service can then be executed (Wada et al., 2012; Zeng et al., 2004).

With more and more applications and computing resources being encapsulated as services and published on the web, how to instantiate the composite service with required quality of services (QoS) is quite crucial. This so-called QoS-aware service composition problem has been a hot research topic in the SOC community. Although many works have been presented for it by adopting different mathematic models and algorithms (Alrifai et al., 2012; Alrifai et al., 2010; Canfora et al., 2005; Lin et al., 2011; Wada et al., 2012; Zeng et al., 2004), they simply attempt to maximize user satisfaction by serving them with the composite service instance of the best QoS. According to this kind of "best-effort" strategy, consumers will all surge to service candidates with the best QoS performance for composition. In this case, these candidates will fall overloaded easily and their advertised / promised QoS can not be guaranteed any more. At the same time, other service candidates with decent QoS performance will not be exploited adequately. It will not only lead to a waste of service resources but also discourage numerous service providers to enter into the service market, even if their services can perform quite well in terms of QoS.

This dilemma originates from that these "best-effort" methods fail to take into account that there