

An Integrated Workbench for Model-Based Engineering of Service Compositions

Howard Foster, *Member, IEEE*, Sebastian Uchitel, *Member, IEEE Computer Society*,
Jeff Magee, *Member, IEEE*, and Jeff Kramer, *Member, IEEE Computer Society*

Abstract—The Service-Oriented Architecture (SOA) approach to building systems of application and middleware components promotes the use of reusable services with a core focus of service interactions, obligations, and context. Although services technically relieve the difficulties of specific technology dependency, the difficulties in building reusable components is still prominent and a challenge to service engineers. Engineering the behavior of these services means ensuring that the interactions and obligations are correct and consistent with policies set out to guide partners in building the correct sequences of interactions to support the functions of one or more services. Hence, checking the suitability of service behavior is complex, particularly when dealing with a composition of services and concurrent interactions. How can we rigorously check implementations of service compositions? What are the semantics of service compositions? How does deployment configuration affect service composition behavior safety? To facilitate service engineers designing and implementing suitable and safe service compositions, we present in this paper an approach to consider different viewpoints of service composition behavior analysis. The contribution of the paper is threefold. First, we model service orchestration, choreography behavior, and service orchestration deployment through formal semantics applied to service behavior and configuration descriptions. Second, we define types of analysis and properties of interest for checking service models of orchestrations, choreography, and deployment. Third, we describe mechanical support by providing a comprehensive integrated workbench for the verification and validation of service compositions.

Index Terms—Service-oriented architecture, composite services, services models, Web services modeling, analysis, validation.

1 INTRODUCTION

As the adoption of a Service-Oriented Architecture (SOA) approach and the more general notion of Service-Oriented Computing (SOC) gains popularity, tool support for the increasing number of standards and complex configuration dependencies is expected. While there are specific tools for certain service aspects, there is currently little to support the engineer in building complex service interactions using complementing standards across the standard spectrum. These standards have been designed to cover the service data requirements, interface descriptions, process requirements, and behavior specifications of collaborating services yet only together will they provide a complete environment for the benefits of services to be realized. Current integrated development environments, such as Visual Studio.NET (for Microsoft.NET), focus on the function or code behind a service (illustrated by the focus of consuming or implementing the service rather than the scope of how that service is expected to be used and composed with other services). In other words, there is a gap

between the provision of tools for building a service (the components and interface of a service) and the interactions that the service will provide or require in the environment that it is used.

Service orchestration languages, such as the Web Services Business Process Execution Language (WS-BPEL) [1], aim to fulfill the requirement of a coordinated and collaborative service invocation specification to support the interactions of a local process with multiple service partners. However, an orchestration alone does not fulfill the requirement of an assured collaboration in cross-enterprise service domains. Participating services must adhere to policies set out to support these collaborative roles in a services architecture with obligations to constrain the interactions between services. While policies are generally considered to be resource access based (e.g., security and access control permissions), obligations are equally important in ensuring that collaboration is conducted in an appropriate manner and that the behavior exhibited by participating clients is suitable for given scenarios. This issue is collectively wrapped up in the term Service Choreography. Recent standards efforts have produced choreography languages, such as the Web Services Choreography Description Language (WS-CDL) [2]. In addition, the design and implementation of service components in this architecture style must support the original policies as defined by the service owner and their enterprise. These interacting services can be constructed using various emerging standards and managed by multiple parties in their domain of interest and as such the task of linking these activities across workflows within this domain is crucial. Therefore, of clear interest is the need to support such engineering tasks as process verification, partner service usability, and other properties to verify the

• H. Foster, J. Magee, and J. Kramer are with the Department of Computing, Imperial College London, Huxley Building, South Kensington Campus, London SW7 2AZ, UK.

E-mail: {howard.foster, j.magee, j.kramer}@imperial.ac.uk.

• S. Uchitel is with the Departamento de Computación, Universidad de Buenos Aires, Pabellón 1, Ciudad Universitaria, Buenos Aires C1428EGA, Argentina, and the Department of Computing, Imperial College London, Huxley Building, South Kensington Campus, London SW7 2AZ, UK.

E-mail: suchitel@dc.uba.ar.

Manuscript received 12 May 2009; revised 21 Oct. 2009; accepted 6 Mar. 2010; published online 28 Apr. 2010.

For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSCSI-2009-05-0123. Digital Object Identifier no. 10.1109/TSC.2010.19.