Search Strategies for Automatic Web Service Composition

Nikola Milanovic, Humboldt University Berlin, Germany
Miroslaw Malek, Humboldt University Berlin, Germany

ABSTRACT

This paper investigates architectural properties required for supporting automatic service composition. First, composable service architecture will be described, based on modeling Web Services as abstract machines supported by formally defined composition operators. Based on the proposed infrastructure, we give several options for achieving automatic service composition by treating it as a search problem. Namely, basic heuristic, probabilistic, learning-based, decomposition and bidirectional automatic composition mechanisms will be presented and compared. Finally, it discusses the impact and outlook for automatic composition.

Keywords: automatic composition; Web Services

INTRODUCTION

Service oriented architectures (SOA) and Web services have been present in the mainstream scientific and industrial focus for many years. SOA promised advances in enterprise integration, B2B interactions and novel ways to process business workflows. However, industry is still using SOA mainly inside an enterprise as a helper for integration of different systems. Native WS capabilities are standardized: communication (SOAP), description (WSDL) and discovery (UDDI) (Papazoglu, 2003). Apart from that, WS architecture stack is mainly empty, meaning unstandardized.

There are many additional WS-frameworks and specifications aspiring to become standards (e.g., WS-Addressing, WS-Transactions, or WS-Coordination). What is not clear, however, is how they can or will cooperate with one another. Each solution targets a specific problem not taking into account other requirements. What is currently missing is a unification effort toward WS-Architecture (Vinoski, 2004). Our goal is identification of key SOA elements and constraints required to support service composition, verification of composition correctness and automatic composition. In this paper, we briefly present our previous
work on architectural concepts and requirements, and focus on the problem of automatic service composition. Although the remaining part of the paper is based on Web services as the most prominent SOA available today, proposed methods are not limited to solving WS-specific issues only, since they offer architectural approach for designing SOA supporting automatic service composition property.

The need for automatic service composition is justified by the ubiquity of the Internet which is forcing enterprises to abandon their heritage business models and legacy systems, and organize themselves into virtual enterprises (Heuvel, 2003). On demand creation of virtual enterprises can shorten delivery times, increase product quality, deliver personalized services, decrease transaction costs, and accommodate short-term cooperating relationships, which can be as brief as a single business transaction. This paradigm requires a shift from tightly coupled business components to more flexible and loosely coupled ones (Webber, 2003) that now dynamically interact with each other through automatic composition in ways that were not predefined and/or predicted in deployment time. The two major attributes required for such environment are extensibility and adaptivity. It is clear that in open environment like this, where services dynamically interact with each other on demand, being able to ensure correctness (dependability, security, timeliness) plays a crucial role. Web service architecture is considered a solution that can support extensibility and adaptivity required for dynamic composition (Yang, 2000).

The rest of the paper is organized as follows: first, our previous work in the area of the composable service architecture and modeling services as abstract machines will be described, that will serve as an environment in which automatic composition will be performed. Then, automatic service composition will be defined as a search problem and relevant properties of the problem will be examined. The state space and equality of abstract machines will be defined before proceeding with the following automatic composition mechanisms: basic heuristic, probabilistic, learning-based, backwards (decomposition) and hybrid. Finally, comparison with related approaches will be given, followed by conclusion and future work.

COMPOSABLE SERVICE ARCHITECTURE

Web service composition, as well as component composition in general, can be observed at two levels: component (service) and architectural level. Our survey of the composition proposals at the component level can be found in Milanovic (2004a). At this level, it is discussed how to orchestrate or choreograph services in different execution patterns using solutions like BPEL (Curbera, 2003; Andrews, 2004) and BPELJ (Blow, 2004), Web component (Yang, 2002; Yang, 2004), semantic Web and OWL-S (Ankolekar, 2002; McIlraith, 2002; Narayanan, 2002), Petri nets (Hamadi, 2003; Zhang, 2004), and finite state machines (Berardi, 2003; Fu, 2002; Bultan 2003). General problem of “industrial” approaches is lack of formal verification mechanisms, while more “academic” approaches are not easily applied in real-world production and enterprise frameworks and some face scalability problems. The issue that has been rarely addressed at all is modeling of non-functional properties, although it has received some attention lately (Zhang 2005).

Modeling service composability at the architectural level is in its embryonic stage and has some roots in architecture description languages (ADLs) (Medvidovic, 2000), which are used to specify high-level compositional view of a software application. ADL focuses on software generation out of deployed components and offers state-transition semantics for analysis and verification of application specification. However, it has been noted (Schmidt, 2001) that new mission-critical and service-oriented applications require additional properties, namely trust and dependability analysis.

Most of the composition approaches are concerned with application level - how to facilitate construction of complex applications from available Web services. We form architectural