Insights into Web Service Orchestration and Choreography

Florian Daniel, Politecnico di Milano, Italy
Barbara Pernici, Politecnico di Milano, Italy

ABSTRACT

As the Web service domain is a fast growing and equally fast changing environment, this paper tries to provide a critical snapshot of currently available standards, particularly focusing on Web service orchestration and choreography. The trend over the last few years in the Web services area firmly points towards seamless business logic integration and inter-enterprise collaboration. In order to reach these business goals, both technological and conceptual advances are required; some already have proven their viability, others still have to be made. Among them, Web service orchestration and choreography are of crucial importance, but still lack a widely agreed on development framework comprising both technological and conceptual aspects. Besides discussing problems and solutions regarding orchestration and choreography of Web services, especially from a conceptual point of view, this paper further tries to highlight mutual dependencies existing among orchestration and choreography.

Keywords: business process management; choreography; orchestration; service composition; service coordination; survey; Web services

INTRODUCTION

When analyzing the current literature on Web services and the main problems the authors focus on, it is possible to identify one main trend towards the adoption of novel and emerging Web service technologies as basis for the next generation of (Web) applications and composite Web services. Flexibly composing different services into composite ones that benefit from the functionalities provided by their single component services, and expose them as higher-level composite services by combining them in a value adding manner, becomes thus of crucial importance.

Web services are driven by the paradigm of the so-called Service-Oriented Architecture (SOA), which describes the relationships that exist among service providers, consumers, and service brokers, and thereby provides an abstract execution environment for Web services. Accordingly, the overall current research addressing service composition is based on tech-
nologies and solutions from the area of *Service-Oriented Computing* (SOC). From their first appearance, SOA and SOC have emerged as key conceptual frameworks for the world of Web services. Interestingly, only few authors (mainly from the industrial sector) mentioned the concept of *Service Oriented Programming* (SOP) (Bieber & Carpenter, 2001). Web service choreography and particularly orchestration actually face the problem of *programming*, rather than the one of *computing*, which is a somewhat abstract concept not easily mappable to the concept of *service*. Within the academic area, maybe Wiederhold, Wegner, and Ceri (1992) already envisioned a SOP-like paradigm when speaking about *Megaprogramming* of large software modules encapsulating business logic at a granularity level comparable to that of today’s Web services, but this was in the early 1990s! Obviously, cutting down the whole research on service composition and related issues to the mere concept of programming would be to simplistic, and we definitely do not intend to narrow it down to such a low level of abstraction. Nevertheless, we think a rough comparison of the two concepts represents a challenging intellectual exercise and allows drawing interesting conclusions.

Just as the advent of *Object-Oriented Programming* (OOP) was based on the notion of *objects* as means for modularizing programming functionality, SOP could be defined as a paradigm that looks at *services* as basic functional modules that can be composed or newly defined, just as it happens with objects in object-oriented programming languages. OOP per se did not suddenly provide revolutionary new programming capabilities with respect to conventional procedural techniques, it rather proved to be a good means for isolation and thus fostered reuse, robustness, and scalability. These factors encouraged the emergence of higher-level concepts like object brokers, Java Beans, object containers, which — and actually it is this what we are interested in — finally enhanced interoperability.

Analogously, current proposals can be interpreted as a transition towards a robust SOP framework. Several Web service standardization bodies are currently addressing issues that can be related to the definition of a proper new programming framework. For example, even if we are already speaking about service composition and seamless inter-enterprise integration, there is still discussion over standardization of other system aspects (e.g., reliable messaging, transaction support...) that have already been solved or are under study in other research areas. And as long as there are no robust and commonly agreed upon standards, real interoperability and composition problems cannot be addressed adequately.

**HANDLING THE COMPOSITION TOOLKIT**

**The Mess with the Right Terminology**

As standards and technologies still have to reach stable definitions, also authors writing about service composition are far from using a commonly agreed on terminology. Peltz (2003) defines *orchestration* as executable business process that interacts with both internal and external Web services, and *choreography* as “…tracks the message sequences among multiple parties and sources — typically the public message exchanges that occur between Web services — rather than a specific business process that a single party executes…”

Alonso, Casati, Kuno, and Machiraju (2004) prefer the terms coordination (protocol) and composition rather than choreography and orchestration. Literally, they clarify “…we will use the term *conversation* to refer to the sequences of operations (i.e., message exchanges) that could occur between a client and a service as part of the invocation of a Web service. We will use the term *coordination protocol* to refer to the specification of the set of correct and accepted conversations…” (p. 199). And “…we refer to a service implemented by combining the functionality pro-