Model-Driven Approaches to Service Composition

C. Dumez, Universite de Technologie de Belfort-Montbeliard, Belfort, France
M. Bakhouya, Aalto University, Aalto, Finland
J. Gaber, Universite de Technologie de Belfort-Montbeliard, Belfort, France
M. Wack, Universite de Technologie de Belfort-Montbeliard, Belfort, France

ABSTRACT

Companies are now making their know-how and information available over the network using Web services. Business-to-Business collaboration through Web service interaction is now a necessary step to better satisfy user requests. The act of combining Web services to achieve a common goal - also known as Web service composition - is a complex issue that should be addressed. Many programming languages were developed to realize interaction between services, such as XLANG, WSFL, and BPEL. However, these languages are meant for the implementation and execution rather than providing a visual representation of the composition. In the past few years, the research community has been trying to tackle this issue by proposing model-driven approaches with the main objective to reduce development time. Some of these approaches are based on formal methods in order to describe, analyze, verify and validate the composition. In other words, applying these methods in design phase helps designers to show explicitly the behavior of Web services, to reason on the composition behavior and verify its properties. In this paper, the authors survey model-driven approaches for service composition. The focus is on surveying and classifying approaches that follow model-driven engineering principles for creating high-level models rather than programming concepts.

Keywords: Formal Methods, Model-Driven Architecture, Service Composition, Web-Based Services, Workflow Patterns

1. INTRODUCTION

Model-Driven Engineering (MDE) is a software development methodology that focuses on creating abstract models rather than computing or algorithmic concepts. A modeling paradigm for MDE should provide models that make sense from the user’s point of view and that are precise enough to serve as a base for implementation (Soley, 2000). In this paper, research approaches relying on models suitable for MDE are classified, presented and compared. Unlike approaches based on computing concepts, such as BPEL, model-driven approaches emphasize the specification of the service composition. Specifying the service interactions in a clear, precisely defined and platform-independent model allows for a better understanding of the composition. Furthermore, these models can be used to generate executable code (e.g., BPEL). A

DOI: 10.4018/ijaras.2013100102
A standard model-driven approach for developing composite Web services is presented in Figure 1.

Model-driven approaches to service composition can be classified into three main categories as depicted in Figure 2 (Dumez et al., 2013). First of all, part of the research community proposes to use concurrent system modeling approaches. Petri nets and process algebras fall into this category. These formal methods are also suited for validating the service composition because of their strong mathematical features. Secondly, other researchers propose to model service composition using state-transition models, such as automata (Beek et al., 2006; Bakhouya & Gaber, 2007). Such models are particularly useful for validation purposes and their graphical representation, using directed graphs, allows for a better understanding of the system. Finally, the rest of the community endorses informal graphical approaches, such as BPMN and UML activities, which are two competing standards supported by the OMG. These models, based on flow chart diagrams, are software industry de facto standards for business process modeling. Such models are more easily understandable and well-known in the software designer community.

To illustrate how each modeling approach can be used to specify the interaction between services, we consider the example described in Figure 3. Throughout this paper, we provide the corresponding representation using the discussed models for this particular example. In this test case, service $S_1$ is called first. Then, an exclusive choice (XOR choice) causes the system to call either $S_2$ alone or $S_3$ and $S_4$ in parallel. Finally, the system will invoke service $S_5$ and finish its execution. This means that two execution paths are possible depending on the result of the exclusive choice: $S_1 \rightarrow S_2 \rightarrow S_5$ and $S_1 \rightarrow (S_3 || S_4) \rightarrow S_5$. In Figure 3, we also provide the equivalent BPEL code for completeness and to remove any ambiguity. The BPEL $<invoke>$ activity is used to call Web services. The control flow is structured using activities, such as $<sequence>$ for sequential execution, $<if>/<else>$ for exclusive choice between branches, and $<flow>$ for executing branches in parallel (i.e. concurrently).

Figure 1. Model-driven engineering of composite web services
BDD-Based Combinatorial Keyword Query Processing under a Taxonomy Model
www.igi-global.com/chapter/bdd-based-combinatorial-keyword-query/46283?camid=4v1a

Appropriate Evolutionary Algorithm for Scheduling in FMS
www.igi-global.com/article/appropriate-evolutionary-algorithm-scheduling-fms/55961?camid=4v1a