Chapter II
Modeling Process-Driven SOAs:
A View-Based Approach

Huy Tran
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Austria

Ta'id Holmes
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Austria

Uwe Zdun
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Austria

Schahram Dustdar
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Austria

ABSTRACT

This chapter introduces a view-based, model-driven approach for process-driven, service-oriented architectures. A typical business process consists of numerous tangled concerns, such as the process control flow, service invocations, fault handling, transactions, and so on. Our view-based approach separates these concerns into a number of tailored perspectives at different abstraction levels. On the one hand, the separation of process concerns helps reducing the complexity of process development by breaking a business process into appropriate architectural views. On the other hand, the separation of levels of abstraction offers appropriately adapted views to stakeholders, and therefore, helps quickly re-act to changes at the business level and at the technical level as well. Our approach is realized as a model-driven tool-chain for business process development.

INTRODUCTION

Service-oriented computing is an emerging paradigm that made an important shift from traditional tightly coupled to loosely coupled software development. Software components or software systems are exposed as services. Each service offers its functionality via a standard, platform-independent interface. Message exchange is the only way to communicate with a certain service.

The interoperable and platform independent nature of services underpins a novel approach to business process development by using processes running in process engines to invoke existing services from process activities (also called process tasks or steps). Hentrich and Zdun (2006) call this kind of architecture a process-driven, service-oriented architecture (SOA). In this approach, a typical business process consists of many activities, the control flow and the process data. Each activity corresponds to a communication task (e.g., a service invocation or an interaction with a human), or a data processing task. The control flow describes how these activities are ordered and coordinated to achieve the business goals. Being well considered in research and industry, this approach has led to a number of standardization efforts such as BPEL (IBM et al., 2003), XPDL (WfMC, 2005), BPMN (OMG, 2006), and so forth.

As the number of services or processes involved in a business process grows, the complexity of developing and maintaining the business processes also increases along with the number of invocations and data exchanges. Therefore, it is error-prone and time consuming for developers to work with large business processes that comprise numerous concerns. This problem occurs because business process descriptions integrate various concerns of the process, such as the process control flow, the data dependencies, the service invocations, fault handling, etc. In addition, this problem also occurs at different abstraction levels. For instance, the business process is relevant for different stakeholders: Business experts require a high-level business-oriented understanding of the various process elements (e.g., the relations of processes and activities to business goals and organization units), whereas the technical experts require the technical details (e.g., deployment information or communication protocol details for service invocations).

Besides such complexity, business experts and technical experts alike have to deal with a constant need for change. On the one hand, process-driven SOA aims at supporting business agility. That is, the process models should enable a quicker reaction on business changes in the IT by manipulating business process models instead of code. On the other hand, the technical infrastructure, for instance, technologies, platforms, etc., constantly evolves.

One of the successful approaches to manage complexity is *separation of concerns* (Ghezzi et al., 1991). Process-driven SOAs use modularization as a specific realization of this principle. Services expose standard interfaces to processes and hide unnecessary details for using or reusing. This helps in reducing the complexity of process-driven SOA models. However, from the modelers' point of view, such abstraction is often not enough to cope with the complexity challenges explained above, because modularization only exhibits a single perspective of the system focusing on its (de-)composition. Other - more problem-oriented - perspectives, such as a business-oriented perspective or a technical perspective (used as an example above), are not exhibited to the modeler. In the field of software architecture, *architectural views* have been proposed as a solution to this problem. An *architectural view* is a representation of a system from the perspective of a related set of *concern*s (IEEE, 2000). The architectural view concept offers a separation of concerns that has the potential to resolve the complexity challenges in process-driven SOAs, because it offers more tailored perspectives on a system, but it