Chapter 9
A Tool Chain for Constructing QoS–Aware Web Services

Bernhard Hollunder
Furtwangen University of Applied Sciences, Germany

Ahmed Al-Moayed
Furtwangen University of Applied Sciences, Germany

Alexander Wahl
Furtwangen University of Applied Sciences, Germany

ABSTRACT

Web services play a dominant role in service computing and for realizing service-oriented architectures (SOA), which define the architectural foundation for various kinds of distributed applications. In many business domains, Web services must exhibit quality attributes such as robustness, security, dependability, performance, scalability and accounting. As a consequence, there is a high demand to develop, deploy and consume Web services equipped with well-defined quality of service (QoS) attributes – so-called QoS-aware Web services. Currently, there is only limited development support for the creation of QoS-aware Web services, though. In this work we present a tool chain that facilitates development, deployment and testing of QoS-aware Web services. The tool chain has following features: i) integration of standard components such as widely used IDEs, ii) usage of standards and specifications, and iii) support for various application servers and Web services infrastructures.

INTRODUCTION

Today, service-oriented architectures (SOA) are a widely used paradigm for structuring distributed enterprise applications. In this context, services are often characterized as reusable units of functionality, which are aggregated to automate a particular task or business process. Based on well-known protocols and standards such as HTTP, SOAP, and WSDL, Web services are the prevailing technology for implementing services. In fact, almost all application servers and enterprise service buses (ESBs) include runtime environments for Web services.

When deploying Web services in the area of so-called mission-critical business applications,
pure Web services are not sufficient. A pure Web service implements the required input/output behavior, but does not address explicitly or implicitly – non-functional requirements. However, at business level quality of service (QoS) attributes such as security, message reliability, availability and scalability are crucial prerequisites to be supported by the deployed service. The importance and necessity of QoS-aware Web services, i.e., Web services with well-defined QoS attributes, has been stressed elsewhere (see, e.g., O’Brien, Merson and Bass, 2007; Ludwig, 2003).

In order to create a QoS-aware Web service, one could encode the required QoS attributes directly in the source code of the service. For example, security aspects such as authentication or authorization could be embedded into the business functionality. However, this approach would bring a couple of disadvantages such as increased maintenance and adaptation costs. Ideally, a QoS-aware Web service would be realized according to the separation of concerns principle (see, e.g., Sommerville, 2004): The Web service implementation encodes the core business functionality, whereas QoS attributes are “outsourced” and can be strictly separated from the service’s source code. QoS attributes should be formally described and flexibly attached to a Web service. This strategy yields the following advantages:

- Reduction of the Web service’s source code complexity.
- Higher degree of flexibility to react on changing QoS requirements.
- Increased reusability of Web services for different deployment settings.

With the WS-Policy specification (W3C, 2007b), there exists a well-known framework for formally describing QoS attributes for Web services. Basic building blocks in WS-Policy are so-called assertions. A single assertion may represent a domain-specific capability, constraint or requirement. Before we discuss the importance of WS-Policy in more detail, let us take a look at development tools for Web services. Integrated development environments (IDEs) such as Eclipse, NetBeans and Visual Studio allow the quick construction of pure Web services and offer respective project types. For example, the NetBeans IDE has a project type “Web Application” and Visual Studio comes with a “WCF Library” wizard. There is, however, only limited support for creating QoS-aware Web services. Typically, the support is restricted to WS-Policy assertions for security and message reliability as defined in the standards WS-Security (OASIS, 2006), WS-SecurityPolicy (OASIS, 2009), and WS-ReliableMessaging (OASIS, 2008). The IDEs do not support other QoS attributes. We also observe that a thorough tool chain for creating arbitrary QoS-aware Web services does not yet exist. Except for security and messaging, QoS attributes are typically hard-coded in the Web service, thus violating the separation of concerns principle. Hence, the above-mentioned advantages, which are good software engineering practices, are not applicable.

In this chapter we will present a complete tool chain for developing QoS-aware Web services. The tool chain can be characterized as follows:

- Integration of standard components such as IDEs.
- Support for the separation of concerns principle.
- Usage of widely accepted standards and specifications.
- Support for different application servers and Web services infrastructures.
- Creation, usage and reuse of not yet supported QoS attributes.
- Easy configurability of QoS attributes.

Constructing a completely new development environment would not be realistic in terms of implementation efforts and would have a negative impact on its acceptance. Therefore, our tool chain will reuse and integrate well-known and widely