Chapter 9


Guadalupe Ortiz
University of Extremadura, Spain

Juan Hernández
University of Extremadura, Spain

ABSTRACT

For the last few years, model-driven architecture, aspect-oriented software development and Web service engineering have become widely accepted alternatives for tackling the design and building of complex distributed applications; however, each of them addresses the principle of separation of concerns from their own perspective. When combined appropriately, both model-driven and aspect-oriented software development complement each other to develop high-quality Web service-based systems, maintaining non-functional properties separate from models to code. This chapter provides a methodology that integrates non-functional properties into Web service model-driven development, increasing the systems’ modularity and thus reducing implementation and maintenance costs.

INTRODUCTION

Service-Oriented Architectures (SOA) provide a successful way of letting distributed applications communicate, in a platform-independent and loosely coupled manner, providing systems with great flexibility and easier maintenance. Web services have become one of the best known implementations of SOA, in which academy and industry try to maintain each specific development aspect independent through the use of encapsula-
Focusing on the main advantages of Web services, we can mention, first of all, at a very high level, that developing applications with Web services results in implemented applications with a very loosely coupled environment; loosely coupled applications are normally easier to scale, manage and extend and less susceptible to errors when modifications are required. At a low level, we can highlight two significant advantages: their interoperability and the use of Internet standard protocols for transport. Interoperability is acquired thanks to the use of XML for message creation and service description, being common to all platforms and service developers. The use of Internet standard protocols, such as HTTP, will permit easier communication through the net, avoiding problems with firewalls and letting us take advantages of its wide infrastructure. Thus, Web services provide a unique technology for the integration of business in and outside firewalls, which allows companies to integrate internal applications, to publish active applications for other companies and to use third-party applications.

On the other hand, the software engineering community is currently moving on to application development based on models rather than technologies. Model-driven development allows us to focus on essential aspects of the system, delaying the decision of which technology to use in the implementation for a later step. Models may be used in multiple phases of development, from the initial system specification to its testing. Each model will address one concern, independently of the remaining issues involved in the system’s development, thus allowing the separation of the final implementation technology from the business logic achieved by the system. Transformations between models enable the automated development of the system from the models themselves; therefore, model definition and transformation become key parts of the process.

Although Web service implementation is properly achieved by development middlewares, non-functional properties have not been given due consideration in the named implementations in such a way that a loosely coupled environment, characteristic from the SOA scope, is maintained. In these systems the functionality which does not perform main but added value system objectives –which is called in this chapter non-functional properties- is scattered and tangled all over the main functionality code. Consequently, there is a lack of tools for earlier stages of Web service development in which non-functional properties and WS-* standards integration are considered. This problem was already solved by other disciplines at code level through the use of aspect-oriented techniques, which allow us to encapsulate and modularize transversal concerns in the systems, thus improving the decoupleness of software systems and reducing maintenance costs. In regard with their models, it is desirable to have a graphical notation which facilitates the developer tasks, especially when integrating the property models into the already existent system models. Thus, when trying to solve the non-functional gap both at code and model levels, we arrive at the use of model-driven development in conjunction with aspect-oriented development in which non-functional properties and WS-* standards integration are considered. This chapter describes a methodology that integrates non-functional properties into Web service model-driven development, where services and properties remain separated during all stages of development -from platform-independent models to code- and property traceability across all these phases is maintained.