Chapter 2.23
A Model-Driven Development Framework for Non-Functional Aspects in Service Oriented Architecture

Hiroshi Wada
University of Massachusetts – Boston, USA

Junichi Suzuki
University of Massachusetts – Boston, USA

Katsuya Oba
OGIS International, Inc., USA

ABSTRACT

Service oriented architecture (SOA) is an emerging style of software architectures to reuse and integrate existing systems for designing new applications. Each application is designed in an implementation independent manner using two major abstract concepts: services and connections between services. In SOA, non-functional aspects (e.g., security and fault tolerance) of services and connections should be described separately from their functional aspects (i.e., business logic) because different applications use services and connections in different non-functional contexts. This paper proposes a model-driven development (MDD) framework for non-functional aspects in SOA. The proposed MDD framework consists of (1) a Unified Modeling Language (UML) profile to model non-functional aspects in SOA, and (2) an MDD tool that transforms a UML model defined with the proposed profile to application code. Empirical evaluation results show that the proposed MDD framework improves the reusability and maintainability of service-oriented applications by hiding low-level implementation technologies in SOA.

INTRODUCTION

A key challenge in large-scale distributed systems is to reuse and integrate existing systems to build new applications in a cost effective manner (Vinoski, 2003; Zhang, 2004). Service Oriented Architecture (SOA) addresses this challenge by improving the reusability and maintainability of distributed systems (Arsanjani, Zhang, Ellis, Allam, & Channabasavaiah, 2007; Bichler & Lin, 2006; Endrei, Ang, Arsanjani, Chua, Comte, Krogdahl, Luo, & Newling, 2004; Foster, 2005; Lewis, Morris, Brien, Smith, & Wrage, 2005; Papazoglou, 2003). It is an emerging style of software architectures to design applications in an implementation independent manner using two major abstract concepts: *services* and *connections* between services. Each service encapsulates the function of a subsystem in an existing system. Each connection defines how services are connected with each other and how messages are exchanged through the connection. SOA hides the implementation details of services and connections (e.g., programming languages and remoting middleware) from application developers. They can reuse and combine services to build their applications without knowing the implementation details of services and connections.

In order to make this vision of SOA a reality, this article focuses on a research issue of increasing the reusability of services and connections and addresses this issue by separating non-functional aspects (e.g., security and fault tolerance) of services and connections from their functional aspects. The separation of functional and non-functional aspects can improve the reusability of services and connections because it allows different applications to use services and connections in different non-functional contexts. For example, an application may unicast messages to a service and another may manycast messages to multiple replicas of the service to improve fault tolerance. Also, an application may send signed and encrypted messages to a service, when the messages travel to the service through third-party intermediaries, in order to prevent the intermediaries from maliciously sniffing or altering the messages. Another application may send plain messages to the service via unsecured connection when the service is hosted in-house. The separation of functional and non-functional aspects can also improve the ease of understanding application design and enable the two different aspects to evolve independently. This results in higher maintainability of applications.

This article describes a model-driven development (MDD) framework for non-functional aspects in SOA. The MDD framework consists of (1) a Unified Modeling Language (UML) profile to model non-functional aspects in SOA, and (2) an MDD tool that accepts a UML model defined with the proposed profile and transforms it to application code (e.g., program code and deployment descriptors). The proposed UML profile allows application developers to graphically describe and maintain non-functional aspects in SOA as UML diagrams (composite structure diagrams and class diagrams). Using the proposed UML profile, non-functional aspects can be modeled without depending on any particular implementation technologies. The proposed MDD tool, called Ark, transforms implementation independent UML models into implementation specific application code.

This article describes design details of the proposed UML profile and demonstrates how Ark transforms an input UML model to application code that runs with certain implementation technologies such as Enterprise Service Buses (ESBs) (Chappell, 2004), secure file transfer protocols and grid computing platforms. Empirical evaluation results show that the proposed MDD framework improves the reusability and maintainability of service-oriented applications by hiding implementation technologies in UML models.