Chapter 9
Leveraging Early Aspects in End-to-End Model Driven Development for Non-Functional Properties in Service Oriented Architecture

Hiroshi Wada
National ICT Australia, Australia

Junichi Suzuki
University of Massachusetts, Boston, USA

Katsuya Oba
OGIS International, Inc., USA

ABSTRACT

In Service Oriented Architecture (SOA), each application is designed with a set of reusable services and a business process. To retain the reusability of services, non-functional properties of applications must be separated from their functional properties. This paper investigates a model-driven development framework that separates non-functional properties from functional properties and manages them. This framework proposes two components: (1) a programming language, called BALLAD, for a new per-process strategy to specify non-functional properties for business processes, and (2) a graphical modeling method, called FM-SNFPs, to define a series of constraints among non-functional properties. BALLAD leverages aspects in aspect oriented programming/modeling. Each aspect is used to specify a set of non-functional properties that crosscut multiple services in a business process. FM-SNFPs leverage the notion of feature modeling to define constraints among non-functional properties like dependency and mutual exclusion constraints. BALLAD and FM-SNFPs free application developers from manually specifying, maintaining and validating non-functional properties and constraints for services one by one, reducing the burdens/costs in development and maintenance of service-oriented applications. This

DOI: 10.4018/978-1-4666-2044-5.ch009

paper describes the design details of BALLAD and FM-SNFPs, and demonstrates how they are used in developing service-oriented applications. BALLAD significantly reduces the costs to implement and maintain non-functional properties in service-oriented applications.

INTRODUCTION

Service Oriented Architecture (SOA) is an emerging style of software architectures to build, integrate and maintain applications in a cost effective manner by improving their reusability (Bichler & Lin, 2006; Papazoglou & Heuvel, 2007; Erickson & Siau, 2008). In SOA, each application is often designed in an implementation independent manner with a set of reusable *services* and a *business process*. Each service encapsulates the function of an application component, and each business process defines how services interact to accomplish a certain business goal. Services are intended to be reusable (or sharable) for different applications to implement different business processes.

In order to retain the reusability of services, it is important to separate non-functional properties of applications (e.g., security and reliability) from their functional properties because different applications use each service in different non-functional contexts (e.g., different security policies) (Wada, Suzuki, & Oba, 2008; Bieberstein, Bose, Fiammante, Jones, & Shah, 2005). For example, an application may transmit 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 messages. Another application may transmit plain messages to the service when it is deployed in-house. Separation of functional and non-functional properties improves the reusability of services in different non-functional contexts.

This paper investigates end-to-end model-driven development (MDD) that manages non-functional properties from high-level business process modeling to low-level configurations of implementation technologies such as transport protocols and remoting middleware. To this end, there exist two major research issues: (1) a lack of adequate strategies to specify non-functional properties in business processes and (2) a lack of adequate methods to manage constraints among non-functional properties.

The first issue is regarding the strategies to specify non-functional properties in business processes. In most of the current practice in separating functional and non-functional properties, non-functional properties are specified on a *per-service* basis (Wada et al., 2008; Amir & Zeid, 2004; Ortiz & Hernändez, 2006; Lodderstedt, Basin, & Doser, 2002; Jürjens, 2002; Nakamura, Tatsubori, Imamura, & Ono, 2005; Soler, Villarroel, Trujillo, Medina, & Piattini, 2006; Vokäc, 2005; Baligand & Monfort, 2004; Wang, Chen, Wang, Fung, & Uczekaj, 2004). However, with a per-service strategy, application developers need to manually ensure that each non-functional property is properly configured in a series of services in an ad-hoc manner because each non-functional property tends to scatter over multiple services simultaneously. For example, when a certain message encryption is consistently required throughout a purchasing business process, developers need to manually specify the encryption property for all services in the business process (e.g., retailers, suppliers, distributors and carriers) one by one. When a change occurs in the required encryption level, developers have to manually examine which services the change affects and carefully implement the change in the affected services. It is tedious, expensive and error-prone to consistently specify, maintain and validate non-functional properties on a per-service basis in a large-scale business process.

The second research issue this paper addresses is a lack of adequate methods to manage constraints