When Parameterized Model Driven Development Supports Aspect Based SOA

Valérie Monfort, Université Paris 1 Panthéon-Sorbonne and Université Paris Dauphine, France

Slimane Hammoudi, ESEO, France

ABSTRACT

Service-Oriented Architectures (SOA) are widely used by companies to gain flexibility. Web services are the fitted technical solution used to support SOA by providing interoperability and loose coupling. Basic Web services are being assembled to composite Web services in order to directly support business processes. However, there is much to be done to obtain a genuine flawless Web service, and current market implementations do not provide adaptable Web service behavior depending on the service contract. This paper proposes two different approaches to increase adaptability of Web services and SOA. The first approach is based on Aspect Oriented Programming (AOP) as a new design solution for Web services. The authors have implemented an infrastructure to enrich services with aspects and to dynamically reroute messages according to changes, without redeployment. The second approach combines Model Driven Development (MDD) and Context-Awareness to promote reuse and adaptability of Web services behavior depending on the service context. Parameterized transformation techniques are proposed to bind context with business logic implemented by a service. The aim is to merge the two approaches to abstract and reduce the technical complexity of aspect based service solution.

Keywords: Adaptability, Aspect Based Services, Context, Meta Modeling, Model Composition, SOA

INTRODUCTION

Economical context impacts companies and their Information Systems (IS). Companies acquire new competitors or develop new business skills, delocalize completely, or parts of their organization. Moreover they are faced with powerful competitors, and they continuously have to develop new products that fit to customer needs, sometimes in as little as 3 months. There, IS is thus regularly confronted with these complex evolutions and should overcome these changes. With Service Oriented Architecture, each application owns a set of interfaces, offering services and masking implementation details. Applications can thus be seen as independent black boxes connected to a middleware such as an Enterprise Application Integration bus (EAI). However, this integration solution does not allow the interconnection of
heterogeneous applications or infrastructures, such as for example distant IS. Web services are the cheapest and simplest technical solution to solve this problem. Since they are based on standards such as XML (eXtensible Markup Language), web services offer interoperability and allow loose coupling. However, we noticed several limitations concerning web services as a very monolithic code encapsulating a variety of different concerns such as business, security or confidentiality. Moreover, it is sometimes required to change a Web service’s code according to new needs, and thus having to redeploy the Web service. Each change is time consuming and during that time, the web service is not available. Nowadays, flexibility to changes is thus not optimal in web services. We have proposed aspect based solutions to gain in code simplicity without having to re-deploy the code and in a “non-intrusive” manner (Hmida, 2006; Tomaz, 2006). We have based our latest approach on extended BPEL (Business Process Execution Language) (Andrews, 2003) and temporized automatons (Alur, 1994; Haddad, 2006), that we prototyped by providing client, and server adaptability. Nevertheless, we are convinced this pragmatic and efficient solution is too complex for non expert users and developers and is difficult to maintain, because it requires strong technical knowledge.

Recently, we have investigated a model driven approach and context awareness to provide developers with the mechanisms that allow them to represent an application in an abstract way (in a model) and then, to automatically generate the corresponding code. We aimed to explore adaptability and flexibility on a service platform using context with the benefits of an MDD (Model Driven Development) strategy (OMG, 2001). These benefits are related to productivity, quality, adaptability and maintenance. We propose to discuss about the relevance of merging these two solutions.

We explain our technical approach based on services and aspects to implement adaptability in SOA and BPEL. Next we show how our first researches on the MDD approach ensure service adaptability while using context mechanisms. We comment on the needs of merging and extending these two research works: aspect based services for SOA and context modeling with parameterized transformation. It launches our future researches in this topic.

ASPECT FOR SERVICE ADAPTABILITY

Web Services and Flexibility

Web services (WS), like any other middleware technologies, aim to provide mechanisms to bridge heterogeneous platforms, allowing data to flow across various programs. The WS technology looks very similar to what most middleware technologies looks like. Consequently, each WS possesses an Interface Definition Language, namely WSDL, which is responsible for the message payload, itself described with the equally famous protocol SOAP, while data structures are explained by XML (Tidwell, 2000). Very often, WS are stored in UDDI registry. In fact, the winning card of this technology is not its mechanism but rather the standards upon which it is built. Indeed, each of these standards is not only open to everyone but, since all of them are based on XML, it is pretty easy to implement these standards for most platforms and languages. For this reason, WS are highly interoperable and do not rely on the underlying platform they are built on, unlike many ORPC (object-oriented remote procedure calls) such as DCOM, Corba, Java RMI. According to a vast majority of industrial leaders, WS is the best fitted technology for implementing Service Oriented Architectures. WSs provide a minimalist mechanism to interconnect different applications. But one fundamental point is the importance of the WSDL being the exact interface of the system. As we said earlier, most of ORPC take a great care of hiding the message layer details from the developer. This approach breaks down when the applications involved do not lay on the same middleware infrastructure, and when interoperability be-
Digital Convergence and Horizontal Integration Strategies

www.igi-global.com/chapter/digital-convergence-horizontal-integration-strategies/47281?camid=4v1a