An Approach Based on Service Components for Adapting Web-Oriented Applications

Soumia Bendekkoum, Software Technologies and Information Systems Department, LIRE Laboratory, Constantine 2 University, Constantine, Algeria
Mahmoud Boufaida, LIRE Laboratory, Constantine 2 University, Constantine, Algeria
Lionel Seinturier, UMR CRISTAL, Lille 1 University, Lille, France

ABSTRACT

Service Oriented Architecture is a software design paradigm of choice for building and integrating distributed Information Systems. The greatest challenge of SOA is to make the system more flexible and adaptable to the enterprise and user environment changes. However, services can change constantly. These changes are produced due to adjustment in structure, e.g., changing service signature, integrating new services into existing business services; in behavior e.g., adding new business rules in simple or composite services; and in interaction schema between the services and the clients. This paper presents a solution based on service component concepts for dealing with changes confined to services and clients in SOA-based applications. It uses service components concepts to define adaptable services that facilitate the extension and the customization of existing services in harmony with service users. In addition, it presents an adaptation service-oriented lifecycle scenario to control service changes in the entire service lifecycle ranging from the announcement to the execution phase.

KEYWORDS
Application Integration, Service Components, Service Evolution, Service Flexibility, Service Management, SOA, Web Service Adaptation, Web Services

1. INTRODUCTION

Services Oriented Architecture (SOA) is one of today’s most popular frameworks, which uses the concept of service as an elementary brick to build complex distributed systems. It provides means for the self-description, announcement, discovery and interaction of services (Erl, 2005). These services are autonomous by definition; complex services may be built by using composition tools.
and mechanisms (Sivasubramanian, 2009). Making Enterprise Information System (EIS) more flexible is one of the main objectives of SOA. This latter permits to design the software system as a set of interacting loosely coupled services so that any change or adjustment to the service during maintenance can be made independently of the other software components. However, this is not sufficient to make the EIS more flexible and more adaptable to the changes imposed by the enterprise itself (e.g. the creation of new products, new services, new applications or processes, and so on), and/or those imposed by the environment (e.g. concurrence, fusion and integration operations and so on). Consequently, in SOA-based applications services changes can be made independently from the other components but service users have no idea about these changes; they are not applied in harmony with the users. This causes execution problems and results in inconsistent data.

As a matter of fact, the autonomy, the reuse and the loose coupling are the key elements to enable flexibility in service-oriented systems (Van Der Aalst, Beisigel, Van Hee, Koning, & Stahl, 2007). The flexibility can really be attained if we can easily create applications and processes, make and control changes and use the new services and/or processes any time without any problem. However, among the different technique solutions, products, tools and technologies for implementing SOA architectures, the objective for which SOA is proposed has been quickly forgotten. Currently, the software architecture is becoming more and more complex and very hard to manage, because the business is becoming more agile and needs to change very often, consequently to dynamically control these changes.

The service flexibility solutions are frequently specified using reusable component-based models (Weinreich & Sametinger, 2001). The first type of component models, which grabs our attention, has power reconfiguration capacities, like DCOM (Distributed Component Object Model) and EJB (Enterprise Java Beans). Other models have high power reconfiguration capacities, like CORBA Component Model (CCM), Java Beans and Fractal component model (Baude et al., 2009; Baude & Legrand-Contes, 2009). These types of component solutions are expensive and tightly-coupled technologies.

Web services are another type of component model, which are accessible by a standard communication protocol SOAP, through their service interfaces described by a standard interface description language WSDL (Alonso, Casati, Kuno, & Machiraju, 2004). It is also possible to compose Web service components using a standard orchestration language like BPEL4WS (Chaari, Laforest, & Flory, 2005). Web services have changed the computing world by using standardized tools. However, decoupling interfaces from applications or using standard protocols to increase service flexibility is a key solution, but it’s not sufficient to make an application more adaptable in such highly dynamic environments.

In the same context, Service Component (Agarwal et al., 2005; Weinreich & Sametinger, 2001; Yang & Papazoglou, 2004) appeared as a hierarchical model component based on SOA and Component-based Software Engineering (CBSE) principles to increase applications’ flexibility. The Service Component Architecture (SCA) standard is a technology for creating services and assembling them in composite applications. It is organized around four sets of specifications: assembly language, component implementations, bindings and policies (Seinturier et al., 2012). They are meant to specify how the SOA services should be implemented and how these services should communicate in an independent manner from heterogeneous platforms, communication protocols and programming languages.

To this end, we propose in this paper an adaptable service-oriented application architecture based on a flexible scenario to meet the most critical situations faced by the service-based applications such as:

1. Problems and errors that may occur in the business process execution, particularly when the uncontrolled changes of services are performed, which cause a failure in the overall business process execution. This can be due to a number of reasons: the change of the internal behavior
A Constraint Programming Approach for Web Log Mining
www.igi-global.com/article/a-constraint-programming-approach-for-web-log-mining/165524?camid=4v1a

Finer Garbage Collection in Lindacap
www.igi-global.com/article/finer-garbage-collection-lindacap/47024?camid=4v1a