DYSCO: A PLATFORM FOR DYNAMIC QOS-AWARE WEB SERVICE COMPOSITION

Dessislava Petrova-Antonova and Sylvia Ilieva
Sofia University, Faculty of Mathematics and Informatics
125 Tsarigradsko shosse Blvd., Sofia, Bulgaria

ABSTRACT
Service-Oriented Architecture (SOA) is one of the most promising trends in software engineering, which makes possible development of distributed systems via loosely coupled services. It provides significant efficiency in the development, deployment and execution of the software based on it and therefore could be applied to a wide range of distributed systems such as enterprise applications, embedded systems, component-based systems, etc. The preferred way to realize SOA is based on web services that are composed in complex business processes to meet specific business goals. In this paper we present a platform for dynamic QoS-aware web service composition, called DYSCO. It supports the whole process of web service composition by automatic generation of Service Level Agreements (SLAs) from client’s QoS requirements, negotiation with web service providers and building of executable business processes that invokes the web services with the highest quality. Providing continuous monitoring, violation detection of SLAs and dynamic reconfiguration, DYSCO guarantees the robustness of the produced business processes, building a clients’ trust.

KEYWORDS
Business process, web service composition, web service.

1. INTRODUCTION
Service-Oriented Architecture (SOA) is a computing paradigm, where all resources are abstracted from hardware and software layers and often implemented as web services. Consuming web services in distributed environment entails a wide range of composition, reconfiguration, negotiation and trust issues that bring new research challenges. Although various approaches and frameworks for web service composition exist, there is still lack of solutions, which produce executable business processes through dynamic negotiation with service providers, adequate Service Level Agreements (SLAs) and automatic compositions of reliable services. Furthermore, there is no answer to the question of what happens with a web service composition when the client requirements for quality of service (QoS) are changed or SLAs are violated. To address these issues, we propose a platform for Dynamic QoS-aware Web Service Composition, called DYSCO.

The goal of DYSCO is to provide capabilities for dynamic composition of web services based on functional and QoS requirements of the clients. Its main objectives are: (1) Dynamic composition of web services and generation of executable business process; (2) Automatic generation of SLAs for each web service of the composition and negotiation with the web service providers; (3) QoS monitoring; (4) SLA violation detection; (5) Dynamic reconfiguration in case of SLAs violations; and (6) Compliance with widely adopted standards and specifications such as Web Service Description Language (Christensen et al, 2001), Web Services Business Process Execution Language (OASIS WSBPEL TC, 2007), Web Services Quality Factors (OASIS WSQM TC, 2011) and Web Service Level Agreements (Ludwig et al, 2003).

After analysis of the previous work on web service composition approaches in Section 2, Section 3 presents an overview of DYSCO. Finally, Section 4 concludes the paper.

2. RELATED WORK
The main goal of the web service composition is to produce an executable business process. Despite the great BPEL engine support of the leading software vendors, most approaches provide formal specification of
composition model described with Hierarchical Task Networks (Sohrabi and McIlraith, 2009), Statecharts (Lecue, 2009), XML nets (Che et al, 2009), Business Process Modeling Notation (Chang and Lee, 2010), AI planning techniques (Zheng and Yan, 2008) etc. and do not produce executable business processes. In contrast Ge et al (2008) present the composition with direct graph, which finally is transformed according to the BPEL syntax specification in order to be executed. The composition platforms presented in (Aiello et al, 2009; Cardellini et al, 2007) also generate executable business processes described in BPEL.

QoS properties have significant role in web service selection process, helping software developers to choose the web services that meets best non-functional requirements of the composition. The current QoS-aware approaches differ in the type and the number of QoS properties of their QoS models. For example, the composition process presented in (Qiao et al, 2009) takes into account only Response time. In contrast, Chang and Lee (2010) propose a complex QoS model that considers quality in three dimensions: Quality of service (Performance, Throughput, Response time and Availability), Quality of contents and Quality of devices. Another QoS model proposed in (Zeng et al, 2004) takes into account five QoS properties including Execution price, Execution duration, Reputation, Reliability, and Availability. While examining the current QoS models, we found that no one of them refers the Web Services Quality Model (WSQM) developed by Organization for the Advancement of Structured Information Standards (OASIS).

SLAs are essential part of the composition process since they guarantee the quality of the composition’s web services and therefore the overall quality of the composition itself. The automatic generation of SLAs for partner web services of the business process is an issue presented by Qiao et al (2009) and Cardellini et al (2007). The QoS specified in the SLAs has to be monitored in real-time to avoid any violation of guarantees. If the provider cannot provide the agreed level of quality, the composition has to be reconfigured. The approach presented in (Qiao et al, 2009) describes the business process including alternative execution path between specific activities. A drawback of the approach is that the number of partner web services increases significantly. The composition framework presented in (Yu and Reiff-Marganiec, 2006) supports monitoring and provide client with ability to choose from several execution plans. Its disadvantage is that the composition plans cannot be assessed according to the QoS properties of the partner web services.

QoS-aware web service composition is an emerging topic in the area of SOA. The current approaches provide partial solutions that are difficult to be applied in a real world scenario, requiring negotiation with different web service providers, SLA governance, compliance with web service standards, etc. In this paper, we propose an architecture supporting QoS-aware web service compositions, which are negotiable, monitorable, reconfigurable, executable and standard compliant.

3. OVERVIEW OF DYSCO

This section presents an overview of DYSCO platform, describing its components and behaviour.

3.1 Components of DYSCO

The platform DYSCO consists of six components: a repository and five tools, implementing the DYSCO functionality. The Composer of the platform is responsible for dynamic composition of web services based on client’s functional and QoS requirements. It generates an executable business process according to WS-BPEL standard in two steps: (1) it generates an abstract business process without partner links, and (2) creates partner links for activities invoking web services.

Discovery agent searches for web services in the repository that fulfil the functional requirements of the composition. It prepares a candidate web service lists for each activity of the abstract business process generated by the Composer. QoS Evaluator selects the web services that best meet QoS requirements of the composition from candidate web service lists. As a result, it prepares a partner web services list, which is needed for generation of executable business process.

The SLA Negotiator is in charge of negotiating exploitation QoS conditions of web services with the providers. It creates SLAs for the partner web services of the composition. The Monitor collects QoS metrics of the partner web services. In case of SLA violation, it notifies the corresponding provider, who must refine the underlying infrastructure in order to ensure the agreed level of quality. If the provider terminates the
SLA, the Monitor sends the identifier of the web service with violated SLA to the QoS Evaluator, which selects a new partner web service using the QoS offers of the providers.

3.2 Behaviour of DYSCO

The behaviour of DYSCO can be clarified by identifying its main actors as well as its workflows according to particular usage scenarios. The main actors of DYSCO are clients and providers. Clients acts with the platform specifying functional and QoS requirements of the composition. They expect to receive an executable business process described in WS-BPEL along with the corresponding WSDL descriptions of the partner web services. Providers publish WSDL descriptions and QoS offers of web services in the repository of the platform and participate in the SLA negotiation as well as in SLA lifecycle maintenance.

The usage scenario in case of new web service composition includes the steps shown in Figure 1.

Figure 1. Usage scenario in case of business process composing

First, the Client specifies the functional and QoS requirements of web service composition (1), which are sent to the Composer (2). Next, the Composer extracts the functional requirements of the composition (3a), sends them to the Discovery agent (4a) and creates an abstract business process. Next, it extracts the QoS requirements of the composition (3b) and sends them to the QoS evaluator (4b). The Discovery agent finds web services that meet functional requirements of the composition (5) and creates candidate web service list for each activity of the abstract business process (6). It sends the candidate web services lists to the QoS evaluator (7). The QoS Evaluator retrieves the QoS offers for the candidate web services from the repository (8) and based on them selects a web service for each activity of the abstract business process. Finally, it generates a partner web services list (9) and sends it to the SLA negotiator (10). The SLA Negotiator generates SLAs for each web service in the composition (11) and sends them to the providers (12). The providers need to verify his underlying infrastructure according to the received SLAs. If they are able to guarantee the required level of quality, they start with the deployment of the partner web services and create notifications for that (13). The notifications are sent to the SLA negotiator (14). Finally, SLA negotiator generates SLAs for each web service in the composition (15) and sends them to the Composer and the Monitor. The Composer creates partner links so that the invoke activities can call the corresponding operations of the partner web services (17). It sends the produced executable business process to the client (18). The Monitor starts to monitor the QoS properties of the partner web services (19).

Due to the limited space of the paper, the usage scenario in case of SLA is not presented.

4. CONCLUSION

While there are quite a number of approaches and frameworks for web service composition to the best of our knowledge we are not aware of any solution, which automates the whole process of web service composition. Such process requires an automatic generation of SLAs from the client requirements, a negotiation with web
service providers to establish desired quality level, monitoring of QoS properties and dynamic reconfiguration in case of SLA violations.

The above requirements to the web service composition process are addressed by DYSCO. The main contribution of DYSCO is the full support of the clients during the building and maintenance of business processes invoking web services. The clients do not need to care about how to find the web services with desired functionality and quality and how to manage their SLAs. DYSCO provides the bare minimum of capabilities required to build robust business processes from high quality web services. It addresses key technological challenges in the web services world: the relatively poor QoS guarantees regarding web services, the limited trust from clients and the lack of support of the business process lifecycle.

ACKNOWLEDGEMENT

The work presented in this paper was supported by the National Scientific Fund of Bulgarian Ministry of Education and Science under Grant No. MU-01-143.

REFERENCES


Cardellini V. et al., 2007. Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes. Proceeding of IEEE Int.Conf. on Web Services, pp. 743-750.


Che H. et al., 2009. Web Service Composition Based on XML Nets. Proc. of Int. Conf. on Systems, pp. 1-10.

Ge J. et al., 2008. Web Services Composition Method Based on OWL. Proc. of Int. Conf. on Computer Science and Software Engineering, China, pp. 74-77.


