A Policy-Based Dynamic Adaptation System for Service Composition

Sara Zatout, Mahmoud Boufaida and Mohamed Lamine Berkane
Laboratory LIRE, University Constantine 2, 25000 Constantine, Algeria
saraz.constantine@gmail.com, mboufaida@umc.edu.dz, mohamed_lamine.berkane@univ-constantine2.dz

Keywords: Service Orchestration, Adaptation, Quality-of-Service (QoS), WS-Policy.

Abstract: As web services are distributed and autonomous applications that live in dynamic environments, any unexpected change to a service such as a QoS parameter degradation, could potentially lead to faults in the composition. Therefore, these parameters need to be identified and controlled during execution. In this paper, we present an architecture that features a plan for monitoring the orchestration processes during execution. This architecture concerns the dynamic adaptation with regard to services composition. The architecture is based on QoS requirements and adaptation strategies expressed with WS-Policy. We illustrate our proposal by means of an example related to a bookstore orchestration process.

1 INTRODUCTION

Web Services (WS) have become a universal technology for the integration of distributed and heterogeneous applications over the Internet (Charfi and Mezini, 2004). Their modeling simplifies the business application development and interoperation (Tsalgatidou and Pilioura, 2002). Services come in two flavors: simple services and composite ones (Papazoglou and Van den Heuvel, 2003). WS composition consists of combining and coordinating a set of services with the aim to achieve a functionality that cannot be achieved by a single WS. An orchestration process represents a form of composition in which the various services can be efficiently organized through a flow to execute a business process. The orchestration must be dynamic, flexible, and adaptable to meet changing business needs (Peltz, 2003). The BPEL (Business Process Execution Language) allows designers to orchestrate individual services, in order to construct higher level business processes, the orchestration specification is expressed in an XML-based language, and it is deployed in a BPEL execution engine (Margaris, 2015). The services involved in a composition have different QoS (Quality-of-Service) criteria.

However, in dynamic environments such as Internet, they may be subject to unexpected malfunctions. Inconsistencies can occur when calling a partner service and performing one of the composition activities, for instance, while an incorrect response arrives or the service is unavailable, and so on. This leads to a deviation from QoS parameters.

Therefore, we assume that the aspects of QoS should be monitored and analyzed at run time, in order to determine if the process meets the defined criteria and requirements.

In this paper, we present a system that allows the monitoring and the dynamic adaptation of services participating in an orchestration, by analyzing aspects of QoS. Our system is composed of two main levels: the monitoring level and the adaptation one. Each of these two levels contains a set of components that interact with each other to ensure a proper functioning of business processes. The first level serves essentially to observe and to detect the QoS deviations of composition processes during the WS execution. It should be able to understand if a given service does not meet certain QoS conditions and requirements. These requirements are generally specified and validated during the design phase. The second level will be used in our system, when a failure has been detected. The adaptation process must be able to identify the type of problem and to adjust it dynamically.

The remainder of this paper is structured as follows. Section 2 presents related work. In Section 3, we present a system on adaptation of business process composition. In Section 4, we give an example to illustrate our proposal. Finally, Section 5 concludes the paper and discusses future work.

2 RELATED WORK

The need to identify and evaluate the quality parameters of service orchestration is an open question receiving attention by many researchers.

In (Erradi et al., 2006), the authors propose a policy-based middleware, called MASC (Manageable and Adaptive Service Compositions) for monitoring WS compositions and dynamically adapting them to various execution changes. However, the adaptation actions that relate to the substitution of services must be dynamically chosen to determine the appropriate alternative service. In the approach, these actions are taken in real time, during the execution phase.

VieDAME (Vienna Dynamic Adaptation and Monitoring Environment for WS-BPEL) (Moser et al., 2008) introduces a mechanism for a dynamic service adaptation of BPEL processes and monitoring QoS attributes. Each service in a BPEL process can be marked as replaceable. Each service and all of its alternative services' endpoints are stored in the VieDAME service repository. A replacement policy can be selected to control which of the available service alternatives will be used. Yet, in this work the authors do not handle the case of updated service repository. This requires a selection again of possible alternative services.

CEVICHE (Complex Event processIng for Context-adaptive processes in pervasive and Heterogeneous Environments) (Hermosillo et al., 2010) is a framework that supports a dynamic business process adaptation, by combining Complex Event Processing (CEP) and Aspect-Oriented Programming (AOP). The information about the processes, contextual environment, business rules, adaptation conditions and alternative services, is saved in an XML file. However, this framework must be referred to a services registry to take account of changes in alternative services.

In (Berkane et al., 2012) the authors propose a pattern-based architecture for designing an adaptation system of business process. However, in this work the adaptation system is specified in the functional layer where the equivalent WS is defined before the process execution.

In our work, the equivalent service is selected during the execution phase.

3 PROPOSED ARCHITECTURE FOR ADAPTING BUSINESS PROCESS ORCHESTRATION

In Figure 1, we present the architecture of our system which allows the dynamic adaptation of WS orchestrations. The orchestration process presents the various services that can be composed efficiently through a flow, in order to execute a business process. The service orchestration is presented in a well-defined language to express the different stages of the composition process. In our work, we choose to use the BPEL language for the

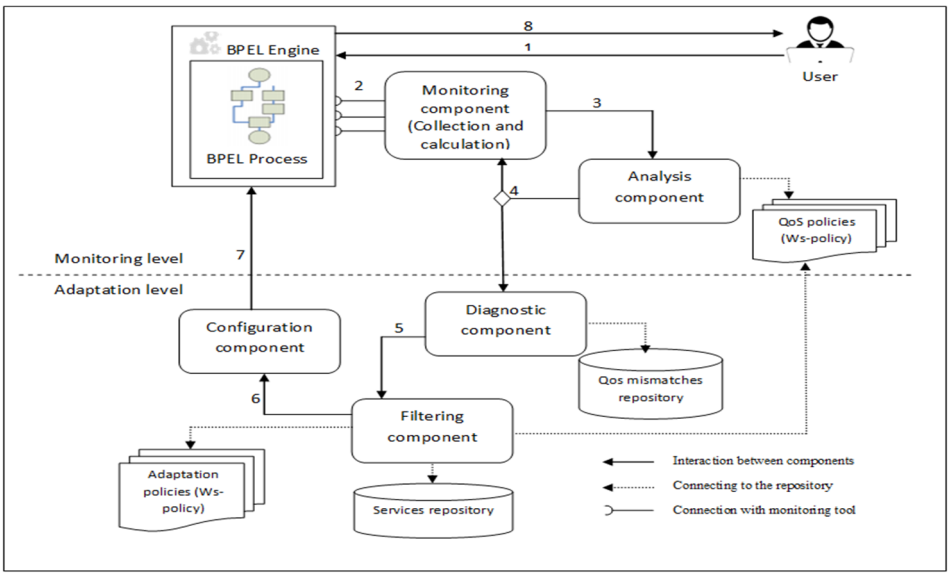

Figure 1: Proposed adaptation architecture.

definition of process models in the form of service orchestrations. BPEL provides an inherent extensibility mechanism at the process and activity level. The orchestration process allows to control the collaboration between services, but unfortunately it does not provide a means to control QoS attributes. Our system focuses on monitoring the different services' orchestration process activities that run in the BPEL engine, then adapting the composition, if it does not meet the requirements set. This system breaks down into two levels: the first level encompasses two components responsible for monitoring and analysis of the composition operation and the second one includes three components that provide a solution for dynamically adapting the composition of services. In the following we present the components of our architecture which is modularized into five distinct components working in cooperation to achieve dynamic adaptation issues.

3.1 Monitoring Component

This component is responsible for providing information about the execution state of the orchestration in the BPEL Engine. It has two parts: the first part is the extraction and collection of QoS values from monitoring tools and the second one aims to calculate the QoS parameters collected in the previous part. After computing the QoS values, the monitoring service sends these parameters to the analysis component for the evaluation.

3.2 Analysis Component

This component is responsible for verifying and evaluating the information collected by the monitoring component to determine the case of an inequality QoS values. It is based on a set of policies expressed with WS-Policy that represents a repository containing the necessary values of QoS that must be verified during the execution of the services. WS-Policy is an extensible model that we can adapt in any system case. In our work, we adopt the QoS policy model suggested in (Mezni et al., 2014), in which a QoS parameter is presented as : <assertion> including attributes <name>, <value> and <unit> which contain other child parameters.

3.3 Diagnostic Component

It is the system responsible for identifying the type of deviation according to the faulty QoS types sent by the analysis component. It interacts with a repository that contains the deviation types of these parameters. Then, it selects the mismatch type, and sends it to the filtering component. The mismatches repository also serves to record the different faulty QoS parameter data. This support can be provided in the specification of the compositions during the design phase.

3.4 Filtering Component

This component's role is to recover the type of deviation sent by the diagnostic component and to select the appropriate configuration action. We provide the adaptation policies (Mezni et al., 2014) and the services repository that allows one to dynamically select the service equivalent to the chosen adaptation action. We use WS-Policy to specify the set of events and actions for each type of event. We adopt the extension AWS-Policy (Autonomic Web Service Policy) presented in (Mezni et al., 2014), in which a model of adaptation policies is described. It contains the actions that can be performed in the case of a detected event, described in an <EventAssertion> element. An adaptation plan can consist of a set of adaptation actions described in an <ActionAssertion> element. However, we will not introduce the name of alternative WS in adaptation policies. The filtering component is responsible for dynamically selecting WS equivalent to the adaptation process.

3.5 Configuration Component

The configuration component is responsible for retrieving the information sent by the filtering component, by applying the necessary changes to the system. It presents a final step in the adaptation process. After the adaptation of the orchestration process, the monitoring system takes the control of the interactions following the composition process with the partner services.

4 EXAMPLE

To provide illustration with regard to our proposed architecture, we take a simple example of a bookstore process (Chan and Bishop, 2009). The process includes a series of activities allowing users to buy books online. The system begins by authenticating the user. After that, the system invokes the "ResultSorting" WS to present the existing catalogues. Then, the user can browse the catalogues for making a search. (S)he can select

items and add them to a (ShoppingCart). The process will run two spots in parallel: one to find the provider of the chosen books; the other one to calculate the price. After that, an order will be generated.

We present an example of the ”ResultSorting” service’s QoS policy. Figure 2 shows the QoS requirements for the “ResultSorting” WS.

```
<wsp:Policy  xmlns:wsp="schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:qosp="garnize:8080/schema/qospolicy"
name="ResultSortingService"
operation="ResultSorting" scope="QOS" location="provider">
<wsp:ExactlyOne>
<wsp:All>
<qosp:All name="performance">
<qosp:All name="responsetime">
<qosp:Assertion name="executiontime" value="1300" unit="ms"/>
<qosp:Assertion name="latency" value="1000" unit="ms"/>
</qosp:All>
<qosp:Assertion name="throughput" value="15" unit="mb"/>
</qosp:All>
<qosp:All>
<qosp:Assertion name="availability" value="70" unit="mb"/>
</qosp:All>
<qosp:All>
<qosp:Assertion name="reliability" value="50"/>
</qosp:All>
<qosp:All>
<qosp:Assertion name="successability" value="85"/>
</qosp:All>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

Figure 2: ResultSorting QoS policy.

The filtering service uses the adaptive policies declared in the repository to determine the possible configuration action for this situation. Figure 3 features an example of QoS adaptation policies, used for the bookstore process.

```
<wsp:Policy  xmlns:wsp="schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:qosp="garnize:8080/schema/qospolicy" name = "adaptationset">
<wsp:ExactlyOne >
<wsp:All>
<qosp:All>
<qosp:All  name ="Event">
<qosp:All >
<qosp:AssertionEvent  name ="timeout"/>
<qosp:AssertionEvent  name ="unavailability"/>
</qosp:All >
</qosp:All >
<qosp:All  name ="Plan">
<qosp:AssertionPlan  name ="substitute"/>
</qosp:All >
</qosp:All>
<qosp:All>
<qosp:All  name ="Event">
<qosp:AssertionEvent  name ="unreliability"/>
</qosp:All>
<qosp:All  name ="Plan">
<qosp:AssertionPlan  name ="reexecute"/>
</qosp:All>
</qosp:All>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

Figure 3: Adaptation policy for the bookstore process.

5 CONCLUSION

In this paper, we have presented a policy-based architecture to dynamically monitor and adapt the QoS parameters of orchestration processes. This architecture is divided into two levels: (a) the monitoring level, which consists of two components, namely the monitoring component and the analysis one; (b) the adaptation level containing three components, namely the diagnostic component, filtering component, and configuration component. We used WS-Policy for expressing QoS requirements and to also express adaptation policies. As future work, we intend to present a method for developing and checking QoS policies during the design phase, and to evaluate our proposal with a more complex case study.

REFERENCES

Berkane, L., Seinturier, L., & Boufaida, M. (2012). A pattern-based architecture for dynamically adapting business processes. In *The Fourth International Conferences on Pervasive Patterns and Applications*, pp. 29-35.

Chan, K. M., & Bishop, J. (2009). The design of a self-healing composition cycle for Web services. In *Software Engineering for Adaptive and Self-Managing Systems. SEAMS'09*, pp. 20-27, IEEE.

Charfi, A., & Mezini, M. (2004). Aspect-oriented web service composition with AO4BPEL. In *Web Services*, pp. 168-182, Springer.

Erradi, A., Maheshwari, P., & Tosic, V. (2006). Policy-driven middleware for self-adaptation of web services compositions. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing* , pp. 62-80, Springer.

Hermosillo, G., Seinturier, L., & Duchien, L. (2010). Using complex event processing for dynamic business process adaptation. In *Services Computing (SCC)*, pp. 466-473, IEEE.

Margaris, D. D. (2015). Advance BPEL execution adaptation using QoS parameters and collaborative filtering techniques. Department of Informatics and Telecommunications, p. 139.

Mezni, H., Chainbi, W., & Ghedira, K. (2014). Extending Policy Languages for Expressing the Self-Adaptation of Web Services. J. UCS, 20(8), pp. 1130-1151.

Moser, O., Rosenberg, F., & Dustdar, S. (2008). VieDAME-flexible and robust BPEL processes through monitoring and adaptation. *In Companion of the 30th international conference on Software engineering*, pp. 917-918, ACM.

Papazoglou, M. P., & van den Heuvel, W. J. (2003). Service-oriented computing: State-of-the-art and open research issues, IEEE.

Peltz, C. (2003). Web services orchestration and choreography. Computer, 36(10), pp. 46-52.

Tsalgatidou, A., & Pilioura, T. (2002). An overview of standards and related technology in web services. Distributed and Parallel Databases, 12(2-3), pp 135-162.