Ensuring Cost-Optimal SLA Conformance for Composite Service Providers

Philipp Leitner

Supervised by: Schahram Dustdar

Distributed Systems Group
Vienna University of Technology
Argentinierstrasse 8/184-1
A-1040, Vienna, Austria
lastname@infosys.tuwien.ac.at

Abstract. For providers of composite services, service level agreements (SLAs) provide a means to guarantee a certain service quality to prospective customers. Usually, violating SLAs is associated with costs. However, the means necessary to ensure SLA conformance also generate costs. Oftentimes, it is therefore optimal from a business perspective to violate certain SLAs sometimes, instead of trying for the high (and expensive) road of always satisfying each one. In this paper we will sketch a framework for the prediction of SLA violations and for determining whether an adaptation of the process makes sense economically. If this is the case adaptation actions are triggered, which adapt the composition on either on instance, structural or environmental level. The ultimate goal is to implement a closed-loop system, which self-optimizes the costs resulting from SLA violations.

1 Introduction

Service-oriented architectures are at their core about the integration of systems. This new paradigm is used by Software-as-a-Service providers, which deliver basic IT functionality such as customer relations management or business intelligence as composite services. One important notion for the seamless integration of such outsourced IT services are agreements about the quality that these services need to provide (QoS), typically defined within legally binding Service Level Agreements (SLAs). SLAs contain Service Level Objectives (SLOs), concrete numerical QoS objectives which the service needs to fulfill. If SLOs are violated, agreed upon consequences (usually taking the form of penalty payments) go into effect. However, fulfilling SLAs can also lead to costs for the service provider (e.g., because the composite service provider needs to use more expensive services itself, or because of the costs inherent to optimizing its service composition). It is therefore not trivial for the provider to decide to what extend the service's SLAs should be fulfilled, or which SLAs should (temporarily) be violated for economical reasons. Even more, these decisions should optimally be automated, to allow for fast reactions to changes in the business environment.

In this overview paper we will present a high-level framework for optimizing adaptations of service compositions with regards to SLA violations. We use techniques from the area of machine learning [1] to construct models allowing the system to predict SLA violations at runtime and decide which adaptation actions may be used to improve overall performance. Adaptation can happen on instance level (for one instance only), on structural level (for all future instances), or on environmental level (e.g., migrating the composition engine to a machine with better hardware). An optimizer component decides if applying these changes makes sense economically (i.e., whether the costs of violating the SLAs are bigger than the adaptation costs). If this is the case the respective actions are applied in an automated way. At its core, this system is a closed-loop self-optimizing system [2], with the target of minimizing the total costs of adaptations and SLA violations for the service provider.

The work described in this paper is currently ongoing. However, some important fundamental work has already been published. In [3] earlier work regarding the monitoring of QoS of Web services is presented. Our work on VRESCo [4] forms the basis for the proposal presented here, providing core services such as support for dynamic rebinding. Finally, in [5] we have presented first results regarding the identification of factors influence of business process performance, which is related to the generation of prediction models. The remaining PhD research will be led by two key research questions: (1) How can the factors that influence the performance of a composition be identified, modeled and analyzed, in order to enable prediction of SLA violations at runtime, and trigger adaptations to prevent these violations? (2) How can the tradeoff between preventing violations and the costs of doing so be best formalized, especially considering that many adaptation actions may be interleaved and combined? Even though first steps exist (see Section 3), to the best of our knowledge, these questions have not been answered sufficiently in literature so far. We plan to validate the outcomes of the thesis using a case study, by showcasing how self-optimization can prevent SLA violations both short- and long-term, and comparing the total costs for the service provider with and without the proposed system. We will consider our work successful if using our system leads to significant financial benefits for the provider, while at the same time reducing SLA violations (even if not all violations are prevented).

The remainder of the paper will be structured as follows. Section 2 contains the main contribution of the paper, a description of a system for cost-optimal adaptation of composite services. In Section 3 we give a brief overview over relevant related work. Finally, Section 4 will conclude the paper.

2 Approach Overview

A high-level overview of our approach is depicted in Figure 1. The system implements an optimization cycle in the Autonomic Computing [2] sense, i.e., it follows the basic steps *Monitor* (monitoring the service composition, i.e., measuring QoS values and process instance data), *Analyze* (generating prediction models), *Plan*

(evaluating based on the generated models, the available adaptation actions and the current SLAs of the provider if there are possible optimizations for the composition), and *Execute* (applying these optimizations). The managed element is the *Service Composition*, while four other components (*Composition Monitor*, *Composition Analyzer*, *Cost-Based Optimizer* and *Adaptation Executor*) implement the autonomic manager.

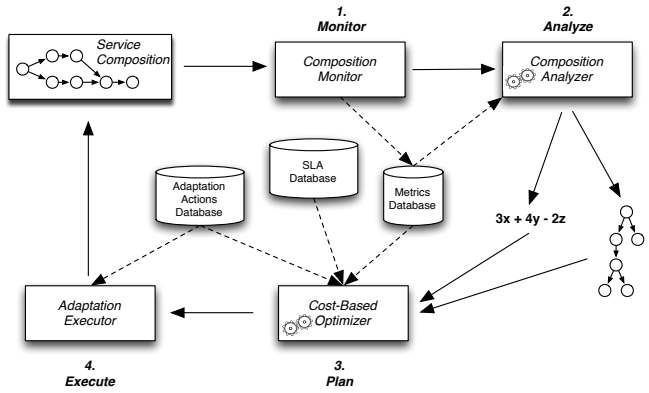

Fig. 1: A Self-Optimizing System for Cost-Optimal SLA Conformance

Our approach is based on the following assumptions: (1) service providers have explicit SLA(s) with their customers, including concrete numerical target values for SLOs and penalty payments for SLA violations; these payments can be staged, i.e., more severe SLA violations can lead to more severe penalties, and (2) there is a database of possible adaptations available, including the costs of these adaptations; costs can be both one-time costs (such as a downtime) or continually (such as increased costs for using more expensive services). The implementation of this database of possible adaptation actions needs to be supported by strong tools, which allow for the generation of the most important actions in a semi-automated way. Costs of adaptation actions can be derived using simulation or analysis of historical data. In the following, we are assuming an autonomous system, however, in some situations the role of the optimizer or executor can also be adopted by a human.

Composition Monitor: The foundation of all work presented here is the ability to gather accurate runtime data. This includes: (1) QoS metrics such as response time or availability of the services used, (2) runtime payload data, such as customer identifiers or ordered items, and (3) technical parameters of the execution environment, such as the availability of the composition engine, or the CPU load of the machine running the composition. The *Composition Monitor* component is used to collect this monitoring data from various sources (e.g., an external

QoS monitor [3] or technologies such as Windows Performance Counters[1]), consolidate it and store it to a metrics database.

Composition Analyzer: The data collected by the *Composition Monitor* is then used to generate prediction models for SLA violations. Prediction models are used to estimate at runtime if a given running instance is going to violate one or more of its SLAs. They are associated with checkpoints in the composition model, at which the prediction is done. Simply put, a prediction model is a function which uses all execution data which is already available at the checkpoint and, if possible, estimations for all missing data, and produces a numerical estimation for every target value in the provider's SLAs as output. In earlier work we have used simple decision trees to implement such models [5], however for future work we investigate the usage of multi-layer perceptrons instead to improve accuracy of predictions.

Cost-Based Optimizer: The *Cost-Based Optimizer* can be seen as the core of the system. This component needs access to all SLAs, as well as a database of possible adaptation actions. The optimizer has to fulfill two important tasks in the system. Firstly, it uses the prediction models generated before to predict concrete QoS values for every running instance and compares the predicted values with the respective SLOs. If SLA violations would occur it checks the Adaptation Actions database for any possible action to prevent the violation, and applies them if it is cost-efficient to do so. This involves solving an optimization problem to decide which combination of actions both prevents most SLA violations and is cheapest to implement. Secondly, if more than a certain threshold of SLA violations (in a given time frame) have been monitored, the component tries to improve the composition itself, i.e., it tries to optimize the composition for every future instance. The main difference is that on composition level more possible adaptation actions exist (mainly because adaptation on this level is less time-critical, so that adaptations which involve e.g., a system downtime are also feasible).

Adaptation Executor: The *Adaptation Executor* is responsible for applying the adaptation actions as planned by the *Cost-Based Optimizer*. Generally, we consider the classes of adaptation actions (action classes) depicted in Figure 2. As discussed before, adaptation can happen either on instance (level 1, i.e., adaptations which affect only a single instance) or structural level (level 2, i.e., adaptations which affect all future instances), and can consist of rebinding base services (R*, i.e., switching from one used service to another), restructuring the composition (S*, e.g., parallelizing some parts of the composition) or adapting the execution environment (E, e.g., upgrading the virtual machine running service composition). Generally, actions of type E always affect all future instances, and are therefore only applicable on level 2.

[1] `http://msdn.microsoft.com/en-us/library/aa373083(VS.85).aspx`

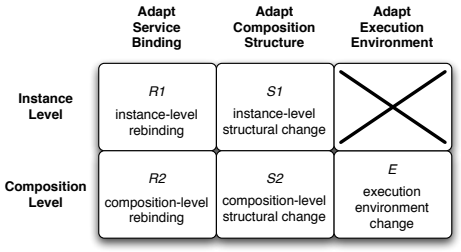

Fig. 2: Classes of Adaptation Actions

Obviously, the concrete execution of adaptation actions depends greatly on the action class. For applying adaptations of type R* we can use the means provided by the execution environment (such as dynamic rebinding as discussed in earlier work [4]). For S* more complex means are necessary, for instance adaptation techniques such as AO4BPEL [6], or the more recent BPEL'n'Aspects [7] approach. Currently, changes of class E are mostly executed manually. However, for some changes of this class automation has been made possible by the recent rise of Cloud Computing. If, for instance, the execution environment is hosted in the Amazon Elastic Compute Cloud[2] it is possible to automatically adapt (some) parameters of the hosting environment via the Amazon S2 API.

3 Related Work

We will now briefly discuss some key related work. QoS monitoring of atomic services is discussed in [3, 8]. Our *Composition Monitor* will partially be based on these results. Monitoring of composition instance data has been discussed in [9]. However, these works do not explicitly cover SLA monitoring, which is the scope of [10]. The authors use an event-based approach to monitor QoS, which is in line with the ideas we have used in [5]. This work also pioneers the idea of SLA impact analysis, which is related to the tasks that our *Composition Analyzer* has to fulfill. Another basic building block of this component is the work presented in [11], which discusses the prediction of QoS (again using an event-based model). Self-adaptation of compositions, another core topic in our work, is discussed in [12]. The MASC system presented there adapts itself in order to recover from failures and improve reliability, however, this system does not try to predict problems and prevent them proactively. Optimizing service compositions with regards to overall QoS is an often-discussed topic, with some seminal work dating back to 2004 [13]. In contrast to this approaches, in our system optimization is done with regard to specific SLOs, which are currently violated, and taking into account the tradeoff between the costs of SLA violations and the costs of adaptation.

[2] `http://aws.amazon.com/ec2/`

4 Conclusions

In this paper we have sketched the architecture of a closed-loop system, which autonomously optimizes service compositions with regard to SLA violations, taking into account the costs caused by the adaptation. Our next steps will be the implementation of an first end-to-end system, which includes prototypes for all four main components of the system. Furthermore, we will define a preliminary model for capturing the costs of adaptation actions.

References

1. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2 edn. Morgan Kaufmann (2005)
2. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36**(1) (2003) 41–50
3. Rosenberg, F., Platzer, C., Dustdar, S.: Bootstrapping Performance and Dependability Attributes of Web Services. In: ICWS '06: Proceedings of the IEEE International Conference on Web Services. (2006) 205–212
4. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: End-to-End Support for QoS-Aware Service Selection, Invocation and Mediation in VRESCo. Technical report, TUV-1841-2009-03, Vienna University of Technology (2009)
5. Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Leymann, F., Dustdar, S.: Monitoring and Analyzing Influential Factors of Business Process Performance. In: EDOC'09: Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference. (2009)
6. Charfi, A., Mezini, M.: AO4BPEL: An Aspect-Oriented Extension to BPEL. World Wide Web **10**(3) (2007) 309–344
7. Karastoyanova, D., Leymann, F.: BPEL'n'Aspects: Adapting Service Orchestration Logic. In: ICWS 2009: Proceedings of 7th International Conference on Web Services. (2009)
8. Moser, O., Rosenberg, F., Dustdar, S.: Non-Intrusive Monitoring and Service Adaptation for WS-BPEL. In: Proceedings of the 17th International Conference on World Wide Web (WWW'08). (2008) 815–824
9. Wetzstein, B., Strauch, S., Leymann, F.: Measuring Performance Metrics of WS-BPEL Service Compositions. In: ICNS'09: Proceedings of the Fifth International Conference on Networking and Services, IEEE Computer Society (April 2009)
10. Bodenstaff, L., Wombacher, A., Reichert, M., Jaeger, M.C.: Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing. (2008) 21–29
11. Zeng, L., Lingenfelder, C., Lei, H., Chang, H.: Event-Driven Quality of Service Prediction. In: ICSOC '08: Proceedings of the 6th International Conference on Service-Oriented Computing. (2008) 147–161
12. Erradi, A., Maheshwari, P., Tosic, V.: Policy-Driven Middleware for Self-Adaptation of Web Services Compositions. In: Middleware'06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware. (2006) 62–80
13. Zeng, L., Benatallah, B., H.H. Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering **30**(5) (2004) 311–327