A Feedback-Aware Binding Component

Anja Strunk

Technical University of Dresden, Institute of Systems Architecture,
Chair of Computer Networks, 01063 Dresden, Germany
anja.strunk@tu-dresden.de

Abstract. The QoS-aware web service composition is an important topic that is intensively discussed in the last ten years. The approach of Canfora, for example, solves this problem in an acceptable time period. The lack of most of the approaches for a QoS-aware composition is, however, that they completely ignore monitoring and user feedback for single web services. In this paper the idea of taking monitoring and user feedback into account while composing web services will be introduced.

Keywords: web service, composition, late-binding, re-binding, feedback-aware.

1 Introduction

As the acceptance of service orientated systems grows, the demand of web services increases. While in the past simple functions like currency conversions were requested, today complex behavior is required. Complex behaviors are realized by *web service compositions* in which web services themselves invoke other web services.

In most cases there is an associated price to pay for use of a composition. Thus the service provider has a financial interest, that his composition is invoked as often as possible. Because of the stark similarities in equivalent services, the provider can only distinguish itself from the competitors by the non-functional properties and the user experience. The adaption on the user's non-functional requirements can be realized by the mechanism called *late binding* – Instead of defining the invoked services at design time, only the needed function and the interface are defined in the form of so called *abstract services*. Directly after the client's request and with the knowledge of the client's QoS-preferences, the abstract services are replaced by concrete ones, which contribute to fulfill the client's non-functional requirements. If a chosen concrete service does not keep its promised properties during execution, it can be replaced by another. Such an adaptive behavior is called *re-binding*.

The set of equivalent services which can replace each other can be obtained by putting a request to the discovery component. Normally the abstract service itself, which describes the needed functionality and interface, is used as request for the service selection.

Beside the QoS properties, considering *monitoring* and *user feedback* during late- and re-binding is also very important. The monitoring feedback informs the composi-

tion service how often a service has broken its promised non-functional properties. The user feedback provides information regarding the user's satisfaction, for example, with regard to the composition's user interface.

If the binding process does not consider the monitoring feedback, low-quality services often violating the promised QoS, could not be identified. Of course, such services are recognized during their execution but cause unnecessary re-binding processes. A Re-binding process may cause a composition process to violate its maximum response time.

User feedback is necessary to optimize the human user's experience using the composition. This in turn directly influences the community's opinion about the composition. The provider is interested in a good opinion, because this will increase the composition's usage.

This paper introduces the idea of taking monitoring and user feedback into account while late- and re-binding. The remainder of the paper is organized as follows: In Section 2, a review about the relevant state-of-the-art in late- and re-binding will be provided; in Section 3, the research problem of considering feedback during binding will be discussed in more detail. Finally, in Section 4 current and future work in developing a QoS- and feedback-aware binding component will be described.

2 Related Work

The Quality-of-Service aware late-binding and re-binding is no new idea. The research community discusses this topic in very detail. The basics are done in the Ph.D of Cardoso [2], which provides the mathematical formulas to calculate time, cost, fidelity and reliability in composed web services. These formulas are extended with further quality of service formulas, such as the one for availability in [1].

Based on the formulas in [2], the different late-binding and re-binding components are built. Yu and Lin in [5] recognize that the main task of developing such components is to solve the optimization problem *OP(S).* OP(S) identifies a composition of concrete services from a set S of possible compositions of concrete services that optimizes the client's requirements.

Zeng et.al in [4] solve the optimization problem through integer programming (IP). They take global as well as local QoS constraints into account. Their approach finds the best composition of concrete services (if it exists), but performance measurements show that the response time of this approach is unacceptably high by a significant number of potential solutions [1].

The second main approach in this area is the one of proposed by Canfora et.al [1]. This approach uses the genetic algorithm. Unlike IP, the genetic algorithm does not find the best solution, but a solution near the best. This is done in favor of a better performance [1]. In order to support multi-attributive optimization, the different QoS criteria are summed up to a global function defining how optimal a composition of concrete services is. By doing so, the individual criteria are not equal anymore. In case a criterion's amplitude is very small compared to the amplitude of the others, this criterion's influence on the optimization's result is less strong. That means, that the

criterion with the highest amplitude dominates the optimization independent of the user's preferences.

The genetic algorithm is faster than the IP approach, although both yield similar solutions. On the other hand, the genetic algorithm has already a huge potential to further optimize the performance. For example, it is well known that the *two-point-cross-over* operator for recombination advances the convergence to a local optimum.

All developed binding components (METEOR-S [10], ASG [11], eFlow [12], WebFlow [13] or SeCSe [3]) are based, more or less, on one of the two algorithms developed by Zeng et al. and Canfora et al. With little need for improvements, they can be used in real-world scenarios.

The main lack of the solutions so far is however, that they neither take monitoring nor user's feedback into account, although there are approaches for reputation model as well as for reputation-based services discovery. For instance Kalepu defines reputation as a function of user ratings, service quality and compliance [14]. Another definition is presented by Maximilian [8]. He proposes a shared reputation model for web services as well as an autonomic service selection based on reputation information. Further reputation-based discovery approaches are proposed by Majithia et al. [15] and Hauswirth [9].

3 Research Problem

Taking monitoring and user feedback into account to optimize web service compositions can be achieved in four steps.

First of all, a feedback model that stores the monitoring and user feedback has to be developed. Perhaps an existing one can be reused or adapted to the binder's circumstances.

The second step is to define mathematical formulas similar to the QoS-criteria, which predict the values of the monitoring and the user feedback of a certain composition of concrete services. The calculated monitoring feedback defines the probability of a re-binding event. The value of user feedback gives an assumption how satisfied the user will be with this composition. Both values are necessary to decide if a certain combination of concrete web services chosen to realize a composition is better than another one.

In the third step the optimization algorithm must be extended so that the solution with the best non-functional properties **and** the best feedback is searched. The genetic algorithm is likely to provide the fastest solution, that's why the approach of Canfora et.al should further be refined.

Finally, after the theoretical treatment of the research problem, the solutions must be implemented and validated. Therefore the common architecture of existing binding components (also called binder) must be extended.

Subsequently, the architecture that supports the four steps will look like the one provided in figure 1. It consists of six main components: discovery, monitoring, negotiation, user's feedback, execution engine and monitoring feedback. While the four components are already proposed by Canfora et al., the remaining two components are newly proposed.

- The discovery component is used to get all the available concrete services for each abstract service.
- The negotiation component is used to document promised non-functional properties as Service Level Agreement.
- The monitoring component is used to recognize re-binding events.
- The execution engine is used to execute the workflow.

The two newly proposed components are required for a feedback-aware binding process.

- The user experience component stores and provides the user's feedback.
- The monitoring feedback component documents how often a concrete web service has broken the promised non-functional properties.

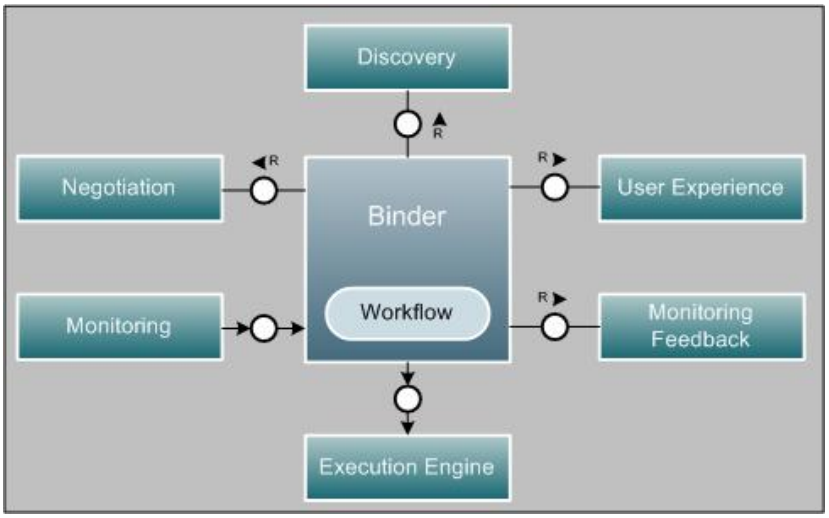

Fig. 1. Rudimental architecture of a binding component, which takes feedback into account.

4 Current and Future Work

Currently, the genetic algorithm approach is being extended to provide a real multi-attributive optimization to solve the binding problem. The extended algorithm will be based on the *pareto optimization*, the classical technique to optimize more than one criterion.

In the future, proposed feedback and reputation models will be evaluated with regard to their fitness into the binding process. Furthermore, a mathematical formula to predict the re-binding probability and the user's from a composition will be derived. Later the integration of these formulas into the genetic algorithm proposed by Canfora will be supported and analyzed.

The genetic algorithm has a large range of control parameters influencing the execution time of the algorithm. That's why as a last step a performance optimization of the genetic algorithm itself must be done. The optimization should identify those values of control parameters, which execute the genetic algorithm in the fastest way.

Finally, a validation and a comparison of the developed solution against the state-of-the-art technologies will be performed.

References

1. Canfora G., Penta D.M., Esposito R., Villani M.L.: An Approach for QoS-aware Service Compostition based on Genetic Algorithms. Genetic and Evolutionary Computing Conference (GECCO'05), Washington, DC, USA (2005)
2. Cardoso J.: Quality of Service and Semantic Composition of Workflows. PhD thesis, Univ. of Georgia (2002).
3. SeSCE Deliverable: A3.D4 Definition of dynamic service binding and (re)negotiation techniques (2005)
4. Zeng L., Benatallah B., Ngu A.H.H.: QoS-Aware Middleware for Web Service Composition. IEEE Transactions on Software Engineering (2004)
5. Yu T., Lin K.: Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. 3rd International Conference on Service Oriented Computing (ICOC'05), Amsterdam, The Netherlands (2005)
6. Cicirello V.A., Smith S.F.: Modeling GA Performance for Control Parameter Optimization. International Journal of Systems Science (1999), vol. 30, page 551-559.
7. Grefenstette J.: Optimization of control parameters for genetic algorithms. IEEE Transaction on Systems, Man and Cybernetics (1986)
8. Maximilien E.M., Singh M.P.: Conceptual Model of Web Service Reputation. Special Section on Semantic Web and Data Management (SIGMOD Record) (2002).
9. Hauswirth L.-J.Vu.M., Aberer K.: QoS-based service selection and ranking with trust reputation management. Federated Conferences (OTM'05) (2005)
10. Verma K., Gomadam K., Sheth A.P., Miller J.A, Wu Z.: C.: The METEOR-S Approach for Configuring and Executing Dynamic Web Processes. Technical Report: http://lsdis.cs.uga.edu/projects/meteor-s/techRep6-24-05.pdf (2005)
11. ASG – Adaptiv Service Grid, http://asg-platform.org/cgi-bin/twiki/view/Public
12. Casatie. F., Shan M.: Dynamic and adaptive compositions of e-services. Information Systems (2001)
13. Greiner E.R.U.: Quality-orientated handling of exceptions in web-service based cooperative process. EAI-Workshop 2004 – Enterprise Application Integration (2004)
14. Kalepu S., Krishnaswamy S., Loke S. W.: Reputation = f(User Ranking, Compliance, Verity), IEEE International Conference on Web Services (ICWS'04) San Diego, California, USA. (2004).
15. Majithia, S., Shaikhali, A., Rana, O., and Walker, D.: Reputation-based Semantic Service Discovery. 13th IEEE International Workshops on Enabling Technologies (WETICE), Modena, Italy (2004).
16. Manokrao U. S., Prabhakar T.V.: Dynamic Selection of Web Services with Recommendation System. International Conference on Next Generation Web Services Practices (NWeSP 2005), Seoul, Korea (2005).
17. Wishart R., Robinson R., Indulska J.: SuperstringRep: Reputation-enhanced Service Discovery. 28th Australasian Computer Science Conference, Australia (2005)