[BSRHSt06] *Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, Ralf Steinmetz;* **An Approach for Replanning of Web Service Workflows;** 12th Americas Conference on Information Systems (AMCIS 2006), Acapulco, August 2006, S.

An Approach for Replanning of Web Service Workflows

Rainer Berbner
Dept. of Computer Science
Technische Universitaet Darmstadt, Germany
berbner@kom.tu-darmstadt.de

Michael Spahn
Dept. of Computer Science
Technische Universitaet Darmstadt, Germany
spahn@kom.tu-darmstadt.de

Nicolas Repp
Dept. of Computer Science
Technische Universitaet Darmstadt, Germany
repp@kom.tu-darmstadt.de

Oliver Heckmann
Dept. of Computer Science
Technische Universitaet Darmstadt, Germany
heckmann@kom.tu-darmstadt.de

Ralf Steinmetz
Dept. of Computer Science
Technische Universitaet Darmstadt, Germany
steinmetz@kom.tu-darmstadt.de

ABSTRACT

The dynamic composition of Web Services to workflows at runtime is a major challenge in the emerging field of Service-oriented computing. At this, preferences and constraints defined by the user have to be considered, e.g. executing a workflow with the cheapest Web Services, but not exceeding a given execution time. Due to the volatile nature of the Web Service environment the actual runtime behavior of Web Services may deviate from the one estimated in the planning phase. Thus, replanning mechanisms are crucial for adapting a workflow to the real behavior ensuring that its execution remains feasible, valid and optimal subject to the preferences and constraints defined by the user. Replanning mechanisms have to perform in real-time not to create further delay to the workflow execution. Thus, in this paper, we introduce a heuristic based replanning mechanism.

Keywords

Web Service composition, Web Service workflows, Quality of Service, Replanning, Optimization.

INTRODUCTION

Due to globalization and deregulation enterprises have to react to dynamically changing international markets and sophisticated customers. As a consequence, agile business processes are a key factor in succeeding in competitive environments. Flexible business processes need architectural support for integrating internal legacy systems as well as for coupling external business partners. The Service-oriented Architecture (SoA) paradigm is recommended as an upcoming architectural blueprint enabling agile business processes (Papazoglou 2003). The basic idea behind this concept is that self-contained loosely coupled services can be composed and orchestrated to cross-organizational business processes. Web Services as a technology based on open XML standards (e.g. SOAP, WSDL and UDDI) have become increasingly important for implementing a Service-oriented Architecture (SoA). With an increasing number of similar Web Services users consider the non-functional properties of a Web Service more intensively (e.g. Menascé 2004). Besides costs, the Quality of Service (QoS) attributes (e.g. availability, response time and error rate) are subsumed as non-functional properties (Cardoso 2002). In our previous work (Berbner et al. 2005a; Berbner et al. 2005b), we have designed and prototypically implemented the proxy-architecture WSQoSX that enables e.g. the selection of specific Web Services at runtime as well as the management of their Service Level Agreements (SLAs). Furthermore, we have designed and evaluated heuristics that calculate an execution plan of a Web Service workflow ensuring that the execution of this plan remains *feasible*, *valid* and *optimal* by considering the preference and constraints defined by the user (Berbner et al. 2006).

However, Web Services do not behave as expected when being executed for two main reasons. First, there is a high unpredictability of the internet as communication channel and the connected servers, with varying network traffic, server loads and downtimes. Second, a further reason is constituted in the assumption of the expected behavior itself. When creating an execution plan, it is assumed that a Web Service behaves as guaranteed by the provider in a previously submitted SLA. Most providers guarantee a behavior which they can fulfill with a very high probability (e.g. 99.9%) even in a worst case scenario. As a consequence, Web Services perform much better under normal conditions, e.g. having a shorter average response time than the one guaranteed by the SLA. Nevertheless, when creating an execution plan, the heuristic has to use the SLA based values, because the resulting execution plan has to be valid with regard to all constraints even in a worst case scenario and not only in an average case.

Due to the deviation of the expected and the real behavior of Web Services, it is necessary to replan the execution plan during runtime to ensure that the execution of a workflow remains *feasible*, *valid* and *optimal*.

If a Web Service is invoked, which is unavailable (e.g. due to server downtime), the workflow is not executable anymore. Thus, the execution plan has to be modified and the current Web Service has to be replaced by another (available) one in order to achieve that the execution of the workflow remains *feasible*.

If the server load of a provider is very high and the response time of a Web Services is much higher than expected, then a constraint, restricting the overall response time of the workflow, may be violated. In this case, the unexecuted slice of the workflow has to be replanned and Web Services having a lower response time have to be used in order to ensure that the current execution of the workflow is *valid* with regard to all constraints.

If all providers have a low or average server load, then the response time of the Web Services is likely to be much lower than guaranteed by their SLA. Each time a Web Service is executed, some response time is saved (compared to the expected behavior). In this case, the execution plan is not optimal anymore because the execution plan could be modified in a way that cheaper Web Services with a higher response time are used, without violating the constraint restricting the overall response time. Thus, the unexecuted slice of the workflow has to be replanned in order to ensure that the current workflow is *optimal* subject to the user's optimization preferences.

However, replanning mechanisms have to perform at real-time otherwise they would delay the workflow execution. As pointed out by Canfora et al. 2005b and Zeng et al. 2004, composing a Web Service workflow with regard to the non-functional behavior of the Web Services involved leads to an optimization problem that is NP-hard. As, the results of Zeng et al. 2004 show, an approach based on linear integer programming is too time consuming for real time scenarios in e-business. Thus, in this paper we present an approach for replanning based on extremely fast heuristics.

The rest of this paper is structured as follows: In Section 2 the composition of QoS-aware Web Service workflows is discussed. Our approach for replanning mechanism of Web Service workflows is proposed in Section 3. The paper closes with a conclusion and an outlook.

WEB SERVICE WORKFLOWS

Web Service composition aims at selecting and inter-connecting Web Services provided by different partners according to a business process (Zeng et al. 2004). Thus, Web Service compositions can be seen as workflows based on Web Services. As depicted in Figure 1, a Web Service workflow consists of abstract tasks i describing the required functionality (e.g. invoking a stock request) of a specific workflow step. The functionality of each task i is provided by m_i different candidate Web Services j ($WS_{i,j}$), which are grouped in category i. Web Services within the same category may differ in their k non-functional parameters ($p_{i,j,k}$). The work in this paper focuses on sequential Web Service workflows because as pointed out by Yu et al. 2005 complex structures (e.g. loops and branches) can be reduced to sequential ones. QoS-aware Web Service composition can be defined as the assignment of specific Web Services to abstract workflow tasks in order to create an execution plan that is optimal subject to the preferences and constraints defined by the user (Berbner et al. 2006). Following this definition, QoS-aware Web Service composition (viz. calculating an optimal execution plan) leads to an optimization problem.

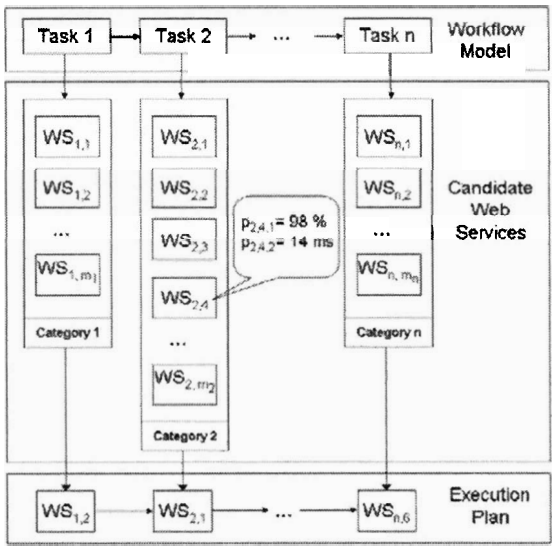

Figure 1. Web Service workflows (Berbner et al. 2006)

We propose a linear model to describe this optimization problem. Table 1 shows how the overall QoS attributes and constraints are calculated. For a more detailed explanation of our model we refer to Berbner et al. 2006.

	Parameter	Overall QoS attributes	Constraints
Additive parameters	$p^{+}_{i,}$	$x^{+} = \sum_{i=1}^{n}\sum_{j=1}^{m_i} p^{+}_{i,j}x_{i,j}$	$x^{+} \le c^{+}$ or $x^{+} \ge c^{+}$
Multiplicative parameters	$p^{\bullet}_{i,j}$	$\approx 1-\sum_{i=1}^{n}\left(1-\sum_{j=1}^{m_i} p^{\bullet}_{i,j}x_{i,j}\right)$	$\ln(c^{\bullet}) \le \sum_{i=1}^{n}\sum_{j=1}^{m_i}\ln(p^{\bullet}_{i,j})x_{i,j}$ or $\ln(c^{\bullet}) \ge \sum_{i=1}^{n}\sum_{j=1}^{m_i}\ln(p^{\bullet}_{i,j})x_{i,j}$
Parameters aggregated by the Min-operator	$p^{\min}_{i,j}$	$x^{\min} = \underset{i=1}{\overset{n}{Min}}\left(\sum_{j=1}^{m_i} p^{\min}_{i,j}x_{i,j}\right)$	$x^{\min} \le \sum_{j=1}^{m_i} p^{\min}_{i,j}x_{i,j} \forall i = 1,\ldots,n$

Table 1. Formulas for calculating overall QoS attributes and defining constraints

The objective function $F(\vec{x})$ expresses the overall utility of the Web Service composition with regard to the user's preferences as a weighted sum of the overall QoS attributes:

$$F(\vec{x}) = \sum_{l=1}^{k^{+}} w^{+}_{l}x^{+}_{l} + \sum_{l=1}^{k^{\bullet}} w^{\bullet}_{l}x^{\bullet}_{l} + \sum_{l=1}^{k^{\min}} w^{\min}_{l}x^{\min}_{l}$$

Each of the k ($k=k^{+}+k^{\bullet}+k^{min}$) QoS attributes is specifically weighted (by w^{+}_{l}, $w^{\bullet}_{l}$, and $w^{\min}_{l}$) to define the importance of the improvement of one unit of the attribute relative to one unit of the other attributes.

As pointed out by Canfora et al. 2005a and Yu et al. 2005, the optimization problem described above is NP-hard. Thus, we concentrate on solving the optimization problem by heuristics. In Berbner et al. 2006 we introduce a heuristic H1_RELAX_IP using a two step approach. First, the LP relaxation of the MIP (Mixed Integer Programming) formulation of the composition problem is solved. In the second step, a backtracking algorithm is used to construct a feasible solution based on the result of the relaxed integer program. The evaluation reveals that H1_RELAX_IP performs extremely fast while generating an almost optimal solution.

A HEURISTIC BASED APPROACH FOR REPLANNING

In this section, a replanning mechanism is discussed, which adapts an execution plan in a way that its execution remains *feasible*, *valid* and *optimal* subject to the preferences and constraints defined by the user.

After having executed a Web Service at position i of an execution plan, the execution plan is divided into two parts as depicted in Figure 2. The first part consists of all positions i' ($i' \leq i$), which already have been executed. The second part consists of all positions i'' ($i'' > i$), which still have to be executed. The k overall attributes of the first part are calculated based on the according parameter values $p_{i,j,k}$ of the used Web Services and the appropriate aggregation functions ($x^+, x^*, x^{\min}$) as discussed in the previous section.

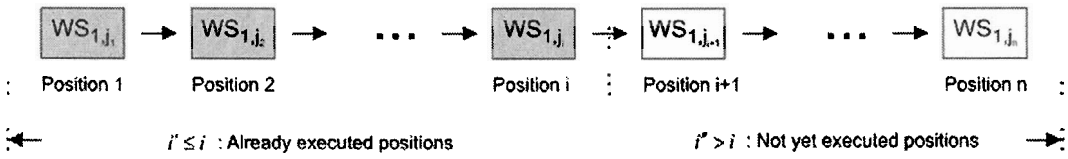

Figure 2. Partitioning of an execution plan

Since all Web Services of the first part have already been executed, the guaranteed parameter values originally taken from a Web Service's SLA are replaced with actual monitored parameter values. The overall attributes of the first part are used to adjust the constraints restricting the overall attributes of the second part to the hitherto existing behavior of the workflow. Using the adjusted constraints, a new optimization problem for the unexecuted part of the execution plan is created and solved using H1_RELAX_IP. This leads to a new execution plan for the second part, which is valid with regard to all current constraints and optimal with regard to the defined optimization preferences. The newly optimized execution plan is used for the further execution of the workflow and is updated using the replanning mechanism each time a Web Service has been executed. Due to the excellent performance of H1_RELAX_IP we do not consider the computation time of the replanning itself.

In the following example (Figure 3) the Web Services are described by two additive parameters: response time ($p_{i,j,1}$) and execution price ($p_{i,j,2}$). The workflow has a single constraint restricting the overall response time (x_1^+) to *2,000 ms* and minimizing the overall execution price (x_2^+) is the only optimization criteria (expressed by $w_1 = 0$ and $w_2 = -1$).

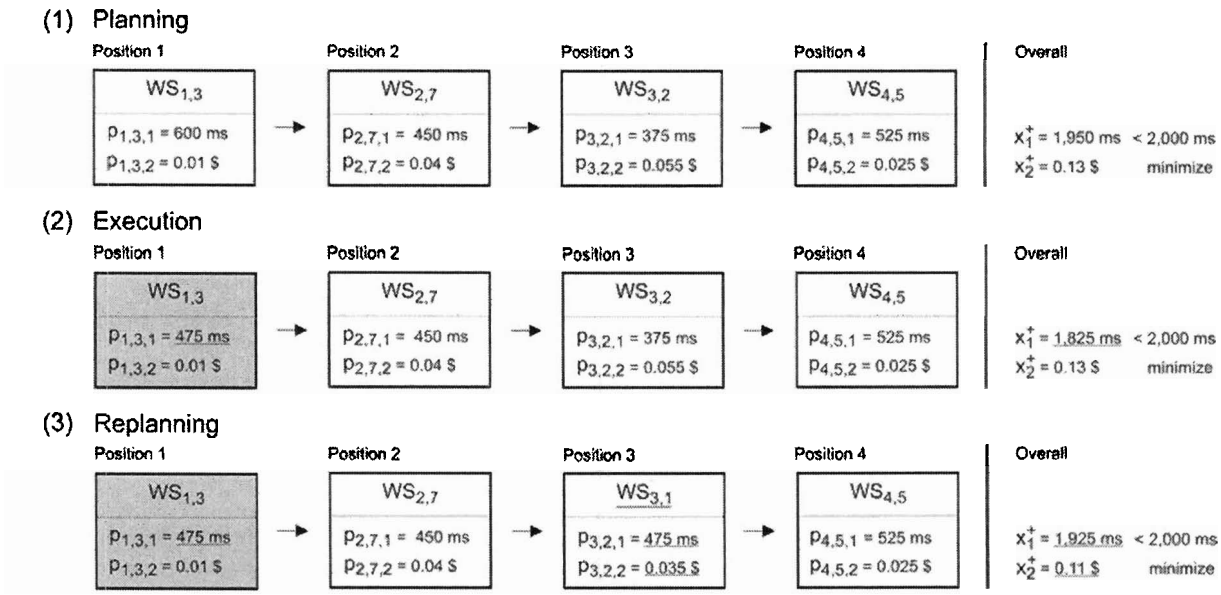

Figure 3. An example for replanning

The heuristic places Web Services at the positions of the execution plan in a way that the overall execution costs are minimized (*0.13$*) and the constraint restricting the overall response time is fulfilled (*1,950 ms < 2,000 ms*). The first Web Service in the execution plan ($WS_{1,3}$) is executed having a lower response time (*475 ms*) than guaranteed by the values taken from its SLA (*600 ms*). If the workflow execution would continue using the current execution plan and every Web Service would behave as expected, then the overall response time would be *1,825 ms*. Nevertheless, this execution plan is not optimal anymore with regard to the optimization criteria of minimizing the overall execution price. To ensure that the execution plan is still optimal, a new optimization problem for the unexecuted part is created. The overall response time of this optimization problem is restricted to *2,000 ms – 475 ms = 1,525 ms*. After solving the new optimization problem using H1_RELAX_IP, an optimized execution plan for the unexecuted part of the workflow is available. Since the first Web Service performed faster than expected, the heuristic was able to replace the Web Service $WS_{3,2}$ at position *3* with the slower but cheaper Web Service $WS_{3,1}$. The resulting execution plan has an overall response time of *1,925 ms* but is cheaper (*0.11$*) than the previous execution plan and is now optimal again with regard to minimizing the overall execution price.

CONCLUSION AND FUTURE WORK

In this paper, we propose a heuristic based replanning mechanism that ensures that the execution of a Web Service workflow remains feasible, valid and optimal even if the actual runtime behavior differs from the one predicted in the planning phase .

Our further research aims at the simulation and evaluation of the approach proposed in this paper. For this, we have started to implement a simulation engine to analyze the trade-off between cost savings and the overhead due to replanning. In this context, we are going to investigate the impact of varying the workflow length, the number of candidate Web Services, and varying the constraints on the potential quality improvement gained by using the replanning mechanism. Furthermore, we will evaluate our approach in business scenarios within the E-Finance industry.

ACKNOWLEDGMENT

The work on this paper is partly sponsored by the E-Finance Lab (http://www.efinancelab.com).

REFERENCES

1. Berbner, R., Grollius, T., Repp, N., Heckmann, O., Ortner, E. and Steinmetz, R. (2005a) An approach for the Management of Service-oriented Architecture (SoA) based Application Systems, Enterprise Modeling and Information Systems Architectures (EMISA 2005), Klagenfurt, Austria, 208-221.
2. Berbner, R., Heckmann, O. and Steinmetz, R. (2005b) An Architecture for a QoS driven composition of Web Service based Workflows, Networking and Electronic Commerce Research Conference (NAEC 2005), Riva Del Garda, Italy.
3. Berbner, R., Spahn, M., Repp, N., Heckmann, O. and Steinmetz, R. (2006) Heuristics for QoS-aware Web Service Composition. Under Submission.
4. Canfora, G., Penta, M.D., Esposito, R., and Villani, M.L. (2005a) An approach for QoS-aware service composition based on genetic algorithms, Genetic and Evolutionary Computation Conference (GECCO 2005), Washington DC, USA, 1069-1075.
5. Canfora, G., Penta, M.D., Esposito, R., and Villani, M.L. (2005b) QoS-Aware Replanning of Composite Web Services, IEEE International Conference on Web Services (ICWS 2005), IEEE, Orlando, FL, USA, 121-129.
6. Cardoso, J. (2002) Quality of Service and Semantic Composition of Workflows, *Department of Computer Science*, University of Georgia, Athens, GA, Ph.D. Dissertation, USA.
7. Menascé, D.A. (2004) Composing Web Services: A QoS View, *IEEE Internet Computing*, 8, 6, 88-90.
8. Papazoglou, M.P. (2003) Service-Oriented Computing: Concepts, Characteristics and Directions, 4th International Conference on Web Information Systems Engineering (WISE'03), IEEE, Rome, Italy, 3-12.
9. Yu, T., and Lin, K.-J. (2005) Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints, 3rd International Conference on Service-Oriented Computing (ICSOC 2005), Amsterdam, The Netherlands, 130-143.
10. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., and Chang, H. (2004) QoS-aware Middleware for Web Service composition, *IEEE Transactions on Software Engineering*, 30, 5, 311-328.