Composition Context for Web Services Selection*

Hong Qing Yu[1], Stephan Reiff-Marganiec[1], Marcel Tilly[2]

[1] *Department of Computer Science, University of Leicester, UK, {hqy1,smr13 }@le.ac.uk*
[2] *European Microsoft Innovation Centre, Aachen, Germany, marcel.tilly@microsoft.com*

Abstract

Often there are several services providing similar functionality, moving the problem of selecting the most suitable to the forefront of interest. In this paper we consider the selection of services in a dynamic environment with changing requirements. In previous work we considered selecting services in isolation, here we present an enhancement to select services in their relation to each other to gain a global optimal solution which nevertheless respects local criteria. Novel contributions are the definition of a composition context and the global multi-criteria optimization mechanism.

1. Introduction

Building software systems by run-time composition of existing Web services is capturing increasing interest in e-Business and e-Government. Meanwhile, many existing Web services are overlapping in functionality and designed for satisfying different Quality of Service (QoS) requirements in different application scenarios. As a result, the competition raises the issue of Web service selection, which in addition to lookup considers finding the best possible service. Our Web service selection research scenario is based on a dynamic context based platform [1] and the criteria for service selection can be gained by reasoning on the context data. The platform also dynamically stores the metadata of registered services and allows updating the QoS at any time. Due to the dynamic aspect, the selection criteria are difficult to predict in advance. From the composition side, the problem becomes more challenging than the single services selection problem, because the composition context needs to be considered to capture the larger context of the service invocation. Moreover, the service selection decision should be made by considering all relevant criteria both for single service selection and composed scenarios. In this sense, aggregating the different related criteria to obtain scores for the competing services in composed scenarios is challenging.

Many projects have studied the QoS driven Web service composition problem. Currently, two kinds of service selection strategies are developed. One focuses on local optimal selection, the other on global optimal selection. **Local optimization** refers to selection methods which only take certain selection constraints related to the current activity in the workflow without specifying and considering the constraints implied by the workflow context and the consequences that the choice will have on later activities. Their biggest advantage is efficiency in selection time, the big drawback is that selections are not necessarily optimal in the global composition context. **Global optimization** on the other hand takes the global selection constraint to select a group of Web services. The key assumption of this strategy is that all suitable Web services for each node have already been discovered and are inside the global optimization search space. Here the selection problem is NP-complete, reducing scalability of the methods. Global optimization approaches do not take into account local criteria, which are often paramount to the user.

We propose a new Web service selection framework which combines the local and global composition contexts to gather the benefits of both approaches. The strategy is what we refer to as a step by step backward knowledge based global optimization; it reduces the complexity and can cope with run-time service faults.

2. Composition Context

We first consider some scenarios to illustrate the concept of composition context. We then make precise what we mean by composition context.

Scenario 1: Error context. For the first activity in a composed service (Mail Service) the selected Mail Service from provider P1 delivers an error but instead P2's Mail Service is working fine. As a result, P2's mail service is selected. This information is tracked in the error context of the composition context and is useful for future executions of the composed service.

Scenario 2: Coordination context. The selection history will also influence the service composition for the later activities in the workflow. When P2's services were selected and executed for activities A1 and A2, then a service from P2 might be the best choice for activity A3.

Scenario 3: Policy context. Looking at various services used within a service composition is useful to make selections dependent on previous selections because of various QoS or SLA constraints. Furthermore the

* This work is partially supported by EU inContext (Interaction and Context Based Technologies for Collaborative Teams) project: IST-2006-034718. The authors would like to thank project partners for discussion.

policy context could also collect 'real' SLA values from the system policies, like response time and availability.

Having seen the scenarios, we come to the first contribution of this paper: a precise description of the composition context (the types are used by the evaluation framework):

Composition Context	Explanation	Type
1. Execution ratings		
1.1 Execution error	The workflow execution engine detected an exception, When the server is invoking.	Numerical
1.2 Coordination error	Two services worked fine independently, however an error appeared during their coordination.	Numerical
1.3 Response time	The time for execution of the service.	Numerical
2. Composition policy		
2.1 Special Cost	This captures special deals between services.	Numerical
2.2 Allowance	This captures which services can and which cannot be used together, or "Is composition allowed?"	Boolean 1 yes 0 no
3. Composition distance		
3.1 Co-location	Are services deployed at the same physical location?	Boolean 1 yes 0 no
3.2 Provider distance	Do services belong to the same provider?	Boolean 1 yes 0 no

We do not claim that the elements defined here are complete, but they have shown sufficient for the scenarios that we studied. If further elements are added to the composition context, then this should not affect the feasibility achieved by applying our selection mechanism – however it might lead to more optimal selections.

3. Web Service Composition Mechanism

We now introduce the novel composition mechanism, which aims for a global optimal solution while respecting local selection criteria. Furthermore, it has polynomial execution time. The main idea is to apply a Backward Knowledge-based Web Service Selection (BKbWSS) approach. In contrast to existing global optimization approaches, the BKbWSS does not need to predict all the global constrains in advance. The BKbWSS approach makes the selection decisions activity by activity based on the currently existing local and global composition context. The composition context is growing as we proceed through the activities. One might argue that the knowledge for selecting the first service is empty and hence we will not select the best one without knowing the forward selection context. While this is true, in practice, it is impossible to predict the execution path as this is going to be influenced by run time data. Hence, the service selection must be based on the user's runtime context when she/he invoke the composition workflow template.

Furthermore, we should not make a decision relying on predicted knowledge which is likely to be wrong. We have to make the service choices based on certain knowledge, which for the selection of the first service is the user's context, for later services the composition context and the user's context. The selection algorithm starts with the selection of the first activity, with the steps below being applied while more activities are encountered in the workflow:

1. Look up the service for the current activity; get the competitive candidate services from registry.
2. Invoke the evaluation framework to rank the candidate services based on the composition context (containing knowledge of previously invoked services, the current candidate services and also user context).
3. Invoke the highest scoring service first. If there is no service available, the execution has failed.
4. If an error occurs when the service is invoked, then record this error information. Return to step 3 to select the next best service. This step adds failure tolerance, which in a distributed setting is essential.
5. If the invocation finishes successfully, log the execution details to the context store. Move to next activity and return to step 1.

The most important step with regard to service selection is step 2. In step 2, the evaluation framework first considers the user's context, then the composition context. Finally all contexts are aggregated to select the best service using the mechanisms presented in [2, 3].

4. Conclusion and Future Work

Selecting the best suitable services to complete a complex composite service is an important research topic. We introduce a Web service selection method combining global composition context with local service selection strategies. The method is based on three parts: the composition context, the selection mechanism (both novel contributions) and the evaluation framework (presented in earlier work [2, 3]).

In the future, we will analyze more composition scenarios and consider completeness of the composition context.

References

[1] inContext project: Unleash Team Power, http://www.in-context.eu, 2007.

[2] S. Reiff-Marganiec, H. Q. Yu and M. Tilly, "Service Selection based on Non-Functional Properties". Non Functional Properties and Service Level Agreements in Service Oriented Computing (NFPSLA07), 2007.

[3] H.Q. Yu and S. Reiff-Marganiec. A Method for Automated Web Service Selection. 2nd International Workshop on Web Service Composition and Adaptation (WSCA08), 2008.