SEMANTIC SERVICES MIGRATION

M. Mohanned Kazzaz

Doctoral Degree Programme (1), FIT BUT

E-mail: xkazza00@stud.fit.vutbr.cz

Supervised by: Jaroslav Zendulka

E-mail: zendulka@fit.vutbr.cz

Abstract: Most of the work on service migration use the same approach which bases on components. These components play roles of searching and selecting the best service to migrate and then take the decision of migration depending on already defined migration conditions. In this paper, we address the problem of service migration between service compositions. We define two ways to apply our approach and propose a solution based on controlling services described semantically.

Keywords: Service composition, Semantic web service, Service migration

1. INTRODUCTION

In cloud computing, web-services are plenty available with their various functionalities. The need for investing their power becomes highly requested to connect these services in some compositions which perform a business process according the user needs instead of spending more time on redeveloping new software. This is addressed by a new approach of service composition with semantic descriptions that provide helpful notation for developers. The developers are able to reuse these services and create new manually or automatically configured services compositions that reduce development time and costs.

With this ability, services requesters and providers are communicating to perform users needs. The user send his request with data he wants to be processed and waits until the provider finish its work and send the response back. However, in the case of request or response messages representing a huge database or set of images needed by an image processing ability, migrating some services to the requester or to some close host to the requester become more useful. The migration of the services allows to reuse network traffic and overhead costs of sending data at least one time via network which may causes connection traffics and depends on network delay and fee.

The rest of this paper is organized as follows: Section 2 introduces the related works about service composition and migration; Section 3 provides a brief description of the OWL-S ontology; Section 4 describes our motivations of this work; Section 5 is the suggested proposal and finally the summary is stated in Section 6.

Our approach is based on using the automatic service composition to 1) re-connect the migrated service with its dependences, 2) create new connections with the host's services and 3) selecting the best service to migrate during the composition phases.

2. RELATED WORK

Web-service composition is addressed in many papers in two main ideas, functional level composition and process level composition.

Web-service composition is performed through four main phases:

Planning, Discovery, Selection, and Execution [1,2].

In the first phase, a work-flow plan of the needed services is designed. In the second phase, services are located from the available services which are registered in UDDI „Universal Description Discovery and Integration”. In the next phase a set of the best candidate services are selected based on non-functional properties. Finally, executing of the whole plan of the selected services are performed.

After getting these fitting services with the desired request, a services description document is produced manually or automatically.

This document is the output of OWL-S „Semantic Markup for Web Services“ or BPEL4WS „Business Process Execution Language for Web Services“ transformation that uses the service composition plan as an input.

Srividya Kona and Gopal Gupta, describe a framework that automatically compose services in all the previous stages depending on user query requirements, and finally generate the OWL-S document [3].

Other approaches describe service migration framework at runtime in cloud. This infrastructure depends on internal components which work together to select the suitable service for migration. The system takes the migration decision depending on the collected information from the environment according to the a proposed migration conditions [4].

3. OWL-S

OWL-S is an ontology that provides a semantic description of web services in the way that make the web-service tasks and composition more easy to achieve in all of its already mentioned phases in automatic way [6].

This ontology describes the service through three main questions: „what the service does“, „how the service works", and „how to access the service“.

Each answer of these questions presents a property of the service, so the properties are „presents“, „describedBy“, and „supports“.

Finally, the service description will consist of three main components: „ServiceProfile“, „Service Model“, and “Service Grounding“.

The service profile provides a helpful information that searching agents needs to determine to find the best service of each request.

The service model describes the process of executing the service.

The service grounding supports the clients with the needed information about how to access this service regarding of communication protocols and messaging formats [7].

4. MOTIVATING SCENARIO

Let us consider that we have two service providers A and B, Figure 1 - and they are composed in many business processes. Now, we want to add a new provider C which periodically needs some service from A, for example to perform „data backup“ which may last for long time transferring data to server. This transferring may cause high network-traffic.

So the suggestion is to migrate the backup service or services to C in order to perform the needed work.

To perform this migration there are two proposed scenarios to achieve it.

The first scenario is to move the desired service to C, which rises a need of modifying the UDDI registry that has the available provided services. The migration should keep the relations and dependences of the whole composition by changing UDDI entries.

The second scenario suggests using of a proxy addressing system that maps the new calls to the new locations which should be considered in the service description.

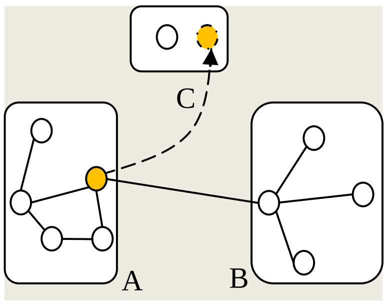

Figure 1: Service composition and migration.

To perform service migration, there are some migration conditions which should be considered before. These conditions can be for example: service status, service movability, connection costs, host descriptions and provider load work.

If two providers request to host an existing service, the system should decide which of them is the best choice as a new location for this service. It may depend on the connections and dependencies costs of it.

In addition to the already mentioned migration conditions, the nature of requesting composition should be taken in consideration. For example, we would like to migrate a service to a real time composition instead of the other compositions in the way that ensure the high quality of service QoS.

5. OUR PROPOSAL

To enable migration of services in Service-Oriented Architecture (SOA), we need to describe two issues: the issue of the initiation of a migration and the issue of the migration itself. The first issue is about how can be set preconditions of a future migration and how can be checked that a system or an architecture meets these preconditions to initiate the migration, while the second issue is about how to migrate a service into a new location in the system or the architecture.

We believe, that the first issue can be solved by usage of formal architecture description languages to describe a state of a system's architecture and by a description of the preconditions of a future migration in a particular formalism (e.g. to describe the system's architecture in the Unified Modelling Language and the preconditions in the Object Constraint Language). Therefore, this paper deals with the second issue of the migration itself.

Let us suppose there is service S, an actual provider P of service S, and two potential providers A and B. We would like to migrate service S from provider P to provider A or provider B, considering different suitability of these providers for future hosting of the service (in our example, let provider A be more suitable for the future hosting of service S than in the case of provider B). Each provider will implement the following special operations:

- int suitabilityOfHosting(SemanticServiceDescription)
- ProviderDescription getProviderDescription()
- void initiateHosting(ServicePackage, ServiceInternalStatus)

Moreover, service S implements the following special operations:

- SemanticServiceDescription getSemanticServiceDescription()

- int suitabilityToBeHostedBy(ProviderDescription)
- ServicePackage getServicePackage()
- ServiceInternalStatus getServiceInternalStatus()
- void attachParallelService(ServiceProviderLocation)
- void finaliseServiceInNewLocation()
- void destroyServiceInOldLocation()

A controller (the controller can be realised as another service of the same SOA where the migrated service belongs) of migration of service S acts as follows:

1. The controller gets the service's semantic description by S.getSemanticServiceDescription() and for each potential provider X, where X in {A,B}, calls X.suitabilityOfHosting(Semantic-ServiceDescription).
2. The controller sorts the provider from the previous step (1) by the return value of operation suitabilityOfHosting(SemanticServiceDescription) in such way that a provider with the bigger return value will be the first (i.e. in our case it will be provider A, see the beginning of this section).
3. The controller gets a description of the first provider from the previous step (2) by its operation getProviderDescription() and pass its return value to service S via S.suitabilityToBeHostedBy(ProviderDescription). If the return value of suitabilityToBeHostedBy(Provider-Description) is positive number, then the service will migrated to the selected provider, otherwise the step (3) will be repeated for a next provider in the order from the previous step (2). If the all return values are not positive numbers, the migration cannot be performed and the controllers quits.
4. Let Y be the provider selected in the previous step (3). The controller calls the service's operations S.getServicePackage() and S.getServiceInternalStatus() and then passes their return values to provider Y by Y.initiateHosting(ServicePackage, ServiceInternalStatus). After that, a new instance of the service is initiated in the location of provider Y with the same internal status as the service's instance in the original location.
5. The controller passes the location on the new service's instance in the original location by S.attachParallelService(ServiceProviderLocation), which will start forwarding of copies of incoming messages to the original location also to the new location (see Section 4).
6. Finally, the controller modifies UDDI registries to respect the new location of the service and calls the service in the new location by S.finaliseServiceInNewLocation() and the service in the original location by S.destroyServiceInOldLocation().

Also, we can provide more service description for each service or information shows times of selecting it to be migrated and its executing times by hosts. This information will help to select the best service to be migrated and the best destination.

Our proposal is an initial proposal of self-adaptive migrating environment by automatically changing locations of services between service compositions. The migrated service may stay at its new location as long as it meets the migration conditions and provide better service performance.

6. SUMMARY

We have introduced an initial service migration framework depending on a controlling service. This service continually collects the information about the participated services and compositions and finally makes the migration decision according to the migration conditions.

ACKNOWLEDGMENTS

This work was supported by the research programme MSM 0021630528 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

We are grateful to our co-researcher Marek Rychlý for the helpful discussions and comments.

REFERENCES

[1] D.B. Claro, P. Albers, and J.K. Hao, Web Services Composition in Semantic Web Service, Processes and Application, Springer, New York, pp. 195-225, 2006.

[2] J. Cardoso, A. Sheth. SemanticWeb Services, Processes and Applications. Springer, 2006.

[3] Srividya Kona, Ajay Bansal, M. Brian Blake, Gopal Gupta: Generalized Semantics-Based Service Composition. ICWS 2008: 219-227

[4] Wei Hao, Tong Gao, I-Ling Yen, Yinong Chen, Raymond Paul: An Infrastructure for Web Services Migration for Real-Time Applications. SOSE '06 Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering, IEEE Computer Society Washington, DC, USA 2006.

[5] Lihua Zheng, Shuang Wu, An Infrastructure for Web Services Migration in Clouds. 2010 International Conference on Computer Application and System Modeling (ICCASM 2010).

[6] The OWL-S Services Coalition. OWL-S Web Service Ontology, 2004. Available from http://www.daml.org/services/owl-s/1.1/.

[7] The OWL-S Services Coalition. OWL-S: Semantic Markup for Web Services, 2003. Available from http://www.daml.org/services/owl-s/1.0/owl-s.html/.