OntoPipeliner: A Semantic Broker-based Manager for Pipelining Semantically-operated Services

Hanmin Jung, Seungwoo Lee, Pyung Kim, Mikyoung Lee, Beom-Jong You

Dept. of Information Technology Research, Korea Institute of Science and Technology Information (KISTI), Republic of Korea
{jhm, swlee, pyung, jerryis, ybj}@kisti.re.kr

Abstract. Like web services, semantically-operated services can be assembled to construct a new composite service. For this, we designed the semantic broker that searches semantic services matched with given conditions, assembles them to dynamically generate pipelines of semantic services, and execute the pipelines. By executing the resulting pipelines, the user can select one which he/she really intended. In this way, our system can help the user who wants to design new semantically-operated services by mashing up the existing semantically-operated services.

Keywords: Semantically-operated service pipelining, Semantic broker

1 Introduction

In this paper, we present a system that assembles existing semantically-operated services to provide a new service. Here, we define a semantically-operated service as a service that always refers to underlying ontology to achieve its predefined goal. That is, a semantically-operated service takes ontology instances as input parameters and computes a result using underlying ontology information. Therefore, it fundamentally differs from traditional web services or semantic web services. Our system can dynamically assemble existing semantically-operated services to help the user to design a new service. Recently, some semantic web service-related technologies, such as OWL-S[1], Semantic Web Services Initiative (SWSI)[2] and Web Service Modeling Ontology (WSMO)[3], have been presented to elaborately assemble existing web services by describing the web services based on ontology. They aim at a guideline and infrastructure for executing web services and designing Upper Ontology which describes properties and capabilities of a web service and agent. However, they do not target semantically-operated services but just traditional web services. This is main difference between our work and the previous ones.

2 Semantically-operated Service Pipelining

[1] http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/
[2] http://www.swsi.org/
[3] http://www.wsmo.org/

Our system is composed of three parts such as *Service Manager*, *Semantic Broker* and *Service Pipelining Tool*. Service manager registers, deletes or edits semantically-operated services in the repository referring to underlying ontology. It supports part of service description defined in OWL-S. It also includes interfaces to help semantic broker to search services. Semantic broker searches and assembles semantically-operated services to generate all possible service pipelines and then execute them. Service pipelining tool helps the user to control the semantic broker and construct a new semantic mash-up. Using this tool, the user can set several conditions such as input instances, output class, visualization type and semantically-operated services or properties required to include. The user also can change input parameters of a service in a pipeline. With executing a pipeline, the tool also interacts with each semantically-operated service in the pipeline.

Our system starts with user's input conditions, which make the tool activate the semantic broker with the conditions. Then, the semantic broker searches semantically-operated services matched with the conditions from the service manager and assembles them to dynamically generate service pipelines, which get back to the tool. For example, let us assume that a user wants to design a service that provides publication network of the most outstanding researcher in a specific research area and has further conditions such as '*neural network*' – an instance of *Topic* class – as an input instance, *Person* as an output class, network as visualization type and '*hasTopicOfPerson*' as an inner property. The input instance constrains semantically-operated services having *Topic* as an input to be the first one in the resulting pipelines. Similarly, the output class and the visualization type constrain semantically-operated services returning instances of *Person* in a form of network to be the last one in the pipelines. The inner property constrains a service using '*hasTopicOfPerson*' property to be included in the pipelines. The semantic broker searches semantically-operated services satisfying these constraints and generates all possible pipelines of the services. As a prototype, our system currently does not support complex control structures such as *split*, *split-join*, *choice*, *if-then-else*, *iterate*, *repeat-while* and *repeat-until*. The resulting pipelines can be executed to help the user to select the best appropriate one. For example, '*getTopicalPeople* + *getPersonNetwork*' and '*getTopicalPeople* + *getTopicalNetwork*' may be two of them. These two reflect the difference in the scope constrained by the given research area. In the former pipeline, the research area does not constrain the researchers except one of the resulting network. On the contrary, in the latter, the research area constrains all the researchers of the resulting network. The user can select one of them which he/she really intended.

3 Conclusion

Our system has been implemented as a prototype and prepared for demonstrating the feasibility of pipelining semantically-operated services at the following URL: http://ontoframe.kr/Broker/main.html. This prototype uses about 100 semantically-operated services, some of which have been developed in OntoFrame2008 – a semantic portal service of academic research information – which has been registered to W3C as a semantic web use case[4].

[4] http://www.w3.org/2001/sw/sweo/public/UseCases/OntoFrame/