Verification of parameters semantic compatibility for semi-automatic Web service composition: a generic case study

Nicolas Lebreton  
Inserm U936  
Faculté de médecine  
35033 Rennes, France  
nicolas.lebreton@univ-rennes1.fr

Christophe Blanchet  
IBCP UMR 5086  
BioSciences Lyon-Gerland  
69367 Lyon, France  
christophe.blanchet@ibcp.fr

Daniela Barreiro Claro  
LASID/DCC/UFBA  
Computer Science  
40.170-110 Salvador-Ba, Brazil  
dclaro@ufba.br

Julie Chabalier  
Natural Solutions  
8 rue Sainte - 13001 Marseille  
www.natural-solutions.eu  
julie_chabalier@natural-solutions.eu

Anita Burgun  
Inserm U936  
Faculté de médecine  
35033 Rennes, France  
anita.burgun@univ-rennes1.fr

Olivier Dameron  
Inserm U936  
Faculté de médecine  
35033 Rennes, France  
olivier.dameron@univ-rennes1.fr

ABSTRACT

We propose an algorithm for checking the semantic compatibility of Web service parameters and for suggesting compatible parameters pairings between two Web services semi-automatically. This algorithm supports semi-automatic Web service composition in workflows where the order of the Web service is known. In our use case, we used the OWL-S semantic description of Web service. We automatically generated a Taverna-compatible Xscufl file whenever possible.

Categories and Subject Descriptors
B.4.0 [Input/Output And Data Communications]: General; I.2.4 [Computing Methodologies]: Knowledge Representation Formalisms and Methods—Representation Languages

General Terms
Verification, Algorithms

Keywords
Web services, semantic compatibility, domain ontology, OWL-S, Xscufl

1. INTRODUCTION

Creating a workflow of Web services is a difficult task [10]. Currently, this is a manual and time-consuming task requiring different expertises. The first step is the selection of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

iiWAS2010, 8-10 November, 2010, Paris, France
Copyright 2010 ACM 978-1-4503-0421-4/10/11 ...$10.00.
2. ALGORITHM

Composing a workflow of Web services requires to define not only the order in which the services are to be called, but also the set of parameters pairings. Each parameters pairing is the association of one a Web service outputs with one of the inputs of the next Web service. Such a pairing can only exist between semantically and syntactically-compatible parameters.

To semi-automatically generate the workflow in Xscufl, we created a script that takes as input a list of OWL-S Web services profiles. It sequentially checks if there is an input and an output that are semantically compatible between Web services.

2.1 Semantic compatibility of parameters

During workflow composition, we assume that the user has already defined the Web services ordering. During each transition from one Web service to the next, we examine all the possible combinations of an output of the first Web service with an input of the next one. Four situations can arise:

- **Identical match**: the input and the output have exactly the same kind;
- **Generalization match**: the input is more general than the output of the previous service;
- **Specialization match**: the input is more specific than the output of the previous service. It is up to the user to make sure that in his conditions of use, the first service will always return results that are semantically compatible with the next service input;
- **Incompatibility**: the input cannot be reconciled with the output of the previous service. This is either because the pairing has to be ruled out, or because the ontology is incomplete.

2.2 Parameters pairings

In our work, four situations can arise at the workflow level:

- the first Web service output is semantically compatible with exactly one input of the next Web service (either through identical match or through generalization match), and no specialization match was detected. The correct pairing can be automatically generated.
- the first Web service output is semantically compatible with exactly one input of the next Web service (either through identical match or through generalization match), but at least one specialization match was also detected. The semantically-compatible pairing is probably the correct one, but the other alternative(s) can be presented to the user;
- the first Web service output is not semantically compatible with any of the inputs of the next Web service (either through identical match or through generalization match), but at least one specialization match was detected. The user should check which (if any) of these alternatives is correct;
- the first Web service output is not semantically compatible with any of the inputs of the next Web service (either through identical match or through generalization match), and no specialization match was detected. The situation cannot be resolved automatically and the user should decide if the problem lies in the workflow itself or in the ontology used to describe the parameters.

3. RESULTS

3.1 Selection of Workflows

Our case study, we select the profile of the three Web services involved in the workflow from the economy domain (author:sciencefictionbook price service:owls, sciencefiction-book publisher service:owls, book publisher service:owls). We used OWLS-TC\(^1\): OWL-S service retrieval test collection. The version three of OWLS-TC consists of indexed OWL-S services from different domains.

3.2 Semantic description of Web services

Domain ontologies provide the semantic information needed by our algorithm for checking the compatibility of the Web services in a workflow. That means knowing what classes describe in detail (the inputs and outputs of Web services). Thus, we know types of Web services parameters. In particular, the OWL-S relation `process:parameterType` describes semantic parameters by linking the `process:parameterType` with classes from domain ontologies. We assume that all Web services share the same domain ontology.

3.3 Generation of the Xscufl file

Once the pairings have been determined at each transition of a workflow, the workflow can be executed. We used the Taverna framework, that relies on Xscufl files. Conceptually, Xscufl is seen as a network of activities related to data and flow controls. Nodes represent the input and output parameters of the Web services of a workflow, and coordination constraints represent relations between the Web services that are not captured by data links. `Arbitrarywsdl` tag defines access to a standard SOAP (Simple Object Access Protocol) based Web service referenced by the URL to a WSDL document and the operation name within that document to access. The `Data source` tags make the links between parameters of Web services and workflow inputs and outputs. The statements `source` and `sink` are used to declare elements of inputs and outputs flows. Figure 1 shows how information from the OWL-S descriptions of the Web services in a workflow is used to generate an Xscufl description.

From the OWL-S file, we supplement tags with necessary data in order to create a workflow compatible with the Xscufl syntax. Properties recovered from OWL-S allow us to properly configure a workflow and to add semantic. Especially, it is possible to retrieve Web service name and its description, to know the WSDL location and to recover the inputs and outputs. The added value is to get classes that describe the inputs and outputs semantics and this will be used to check the input/output compatibility between various Web services.

In our work, the script allows us to retrieve information of our three Web services: we recover service names, operations, WSDL locations, Web services parameters and semantics related. Semantic descriptions allows us to compute input/output compatibility. The figure 2 shows how the different type of matching is done for our case study.

\(^1\)OWLS-TC: http://semwebcentral.org/projects/owls-tc/
In our example, the first Web service (author sciencefictionbook price service.owls) takes an author as input and gives a science fiction book and a price as results. This service returns science fiction book written by the given author. The second Web service (sciencefictionbook publisher service.owls) yields a publisher from a science fiction book. This service returns publisher of a certain science fiction book. The third Web service (book publisher service.owls) yields a publisher from a book. This service returns publisher of a certain book. Our algorithm searches if there is an identical match between these Web services. In this case, the first Web service can be combined with the second Web service because they have the ScienceFictionBook class in common. Moreover, the algorithm will search the superclasses of the output semantic description using the domain ontology (book.owl in our case study). The super class of the ScienceFictionBook is a Book, it matches with the first input of the first Web service so the connection is feasible by a generalization match.

Finally, using our approach we get workflows with a well defined semantic compatibility between Web services inputs and outputs. We find out our case studies setting correctly.

4. DISCUSSION

Technologies based on Web services are becoming increasingly useful for data analysis through the deployment of computational tools. One of the advantages of using Web services is sharing of methods through Internet and their combination to produce complex analysis. Nevertheless, building a workflow to enhance data processing must take care of compatibility problems. Therefore, taking into account the data semantics for binding Web services parameters is a way to facilitate the user composition.

There are many Web services composition standards [5]. Approaches like WSMO [6], OWL-S [8] and SAWSDL [4] use ontologies to describe Web services and add additional meaning to all entities handling by Web services. With the addition of semantics to describe Web services, it is easier to handle and understand their actions, but parameters compatibility problems remain. In this paper, we used OWL-S ontology to describe our Web services. As there is little information on semantics data and also that this information is not available at workflow building time, it is not easy to properly configure Web services. By using OWL-S file and their related domain ontologies, we use semantic descriptions to Web services. We selected OWL-S because it is an OWL-based Web service annotation formalism which made integration with existing domain ontologies in OWL easier. Then we imported domain ontologies. Ontologies allow semantic Web services descriptions by defining the nature and type of inputs and outputs. However, the annotation by ontologies is static and manual.

Workflow systems have become a necessary tool for many applications, enabling the composition and execution of complex analysis on distributed resources. Previous work [2] proposed a Web services composition method based on OWL ontology, and designed a system model for services composition. Web services are modeled based on OWL ontology. The services are semantically matched and composed by artificial intelligence techniques. However, it would be interesting to take into account the OWL-S semantic standards to better define the Web services and turn them into a workflow language that is either reusable regardless of the chosen architecture.

There are also other works on the Web services composition that have a narrower field of study. Some systems have primarily focused on supporting the Life Sciences community (biology, chemistry and medical imaging) with workflows monitoring services. Some solutions are proposed for BioMoby Web services composition [9]. DiBernardo et al. present a prototype workflow assembly client that reduces the number of choices that users have to make by restricting the overall set of services presented to them and ranking services so that the most desirable ones are presented first [1].
Most BioMoby clients\(^2\) use the Application Programming Interface (API) in the Web interface, allowing users to start with some data, and then discover services that uses that piece of information, followed by the service discovery that uses the output of the previous service. All these solutions are only valid for BioMoby Web services.

We applied our generic algorithm by using the Xscufl file (Taverna workflow system). We use semantic by checking if there are inputs and outputs that are compatible, either by an identical or a generalization or a specialization match. As discussed above many projects focus on the identical match but we use the hierarchy of the ontology to provide a generalization and a specialization match during the composition. We don’t use approximate matching because classes found nearby may have a meaning close to the desired settings, but the choice of this parameter must be exact. This will have a negative impact on Web service composition, that’s why we disregard this solution.

By comparing our automatically generated workflows with the manual version of the workflows, the advantage of our solution is that setting compatibility does not require absolute knowledge of users. Knowing that information about knowledge is often little available. Semantic description of Web services features is made explicit in the OWL-S file, our approach makes subsequence of the Web service easier. Furthermore, description of Web services included in workflows can be used for other sequences instead of an ad hoc solution.

5. CONCLUSION

This work confirms that checking compatibility of a workflow is possible using existing technologies. We have demonstrated the contribution of ontologies for improving semantic descriptions and checking compatibility during Web services composition. These ontologies are reusable for the achievement of other scenarios, unlike scripts written by hand. Our semantic compatibility algorithm was only tested on linear workflow. Its generalization to more complex control structure remains to be studied.

6. REFERENCES


\(\text{\footnote{\text{BioMoby clients available at http://www.biomoby.org/}}}\)