Addressing the Issue of Service Volatility in Scientific Workflows

Khalid Belhajjame

School of Computer Science
University of Manchester
Oxford Road, Manchester, UK
Khalid.Belhajjame@cs.man.ac.uk

Abstract. Workflows are increasingly used in scientific disciplines for modelling, enacting and sharing *in silico* experiments. However, the reuse of an existing workflow is frequently hampered by the volatility of its constituent service operations. To deal with this issue, we propose in this paper a set of criteria for characterising service replaceability using which substitute operations can be automatically located for replacing unavailable ones in workflows.

1 Introduction

The wide adoption of web services, as a means for delivering both data and computational analysis, together with the use of workflow technology, as a mechanism for loosely aggregating services, has dramatically revolutionised the way many scientists conduct their daily experiments. Using a workflow, a scientific experiment is defined as a series of analysis operations connected together using links that specify the flow of data between them. Enacting the specified workflows allows scientists to gather evidence for or against a hypothesis or demonstrate a known fact. Once tried-and-tested, the specifications of scientific workflows can be, just like with web services, stored in public repositories to be shared and reused by other scientists. For example, the *myExperiment* project[1], launched recently, aims to provide a public repository for sharing workflows (and thus the experiments these workflows model) between life scientists.

In practice, however, the reuse of an existing workflow is frequently hampered by the fact that certain of its constituent service operations are no longer available. Because of this, the workflow cannot be executed nor used as a building block for composing new experiments. This is not surprising; the service operations composing the workflows are supplied by independent third party providers and there is no agreement between service providers and users that compel the providers to continuously supply their services.

A solution that can be adopted to deal with the issue described above would consist in substituting each of the unavailable operations with an operation that is able to fulfil the same role as the unavailable one within the workflow. This

[1] http://myexperiment.org/

B. Krämer, K.-J. Lin, and P. Narasimhan (Eds.): ICSOC 2007, LNCS 4749, pp. 377–382, 2007.

raises the key question as to *what operation is suitable for substituting a given unavailable one*. In this paper, we formally characterise service operation replaceability in workflows. We begin (in Section 2) by presenting the semantic annotations of web services that are used for characterising service replaceability. We then present the criteria that we identified for characterising service replaceability in Section 3 and conclude in Section 4.

2 Semantic Annotations for Characterising Replaceability

For the purposes of this work, we use the workflow definition that we adopted in an earlier work for the detection of mismatches [4]. We define a scientific workflow *wf* as a set of operations connected together using data links. Formally $wf = \langle nameWf,\ OP,\ DL \rangle$, where *nameWf* is a unique identifier for the workflow, *OP* is the set of operations from which the workflow is composed, and *DL* is the set of data links connecting the operations in *OP*.

A service operation is associated with input and output parameters. A parameter is defined by the pair $\langle op,\ p \rangle$, where *op* denotes the operation to which the parameter belongs and *p* is the parameter's identifier (unique within the operation). An operation parameter is characterised by a data type. We assume the existence of the function *type()*, which given a parameter returns its data type.

A data link describes a data flow between the output of one operation and the input of another. Let *IN* be the set of all input parameters of all operations present in the workflow *wf* and *OUT* the set of all their output parameters. The set of data links connecting the operations in *wf* must then satisfy, $DL \subseteq OUT \times IN$. In the rest of the paper, we use *OPS* to denote the domain of service operations, *INS* the domain of input parameters and *OUTS* the domain of output parameters.

To be able to locate the service operations able to substitute an unavailable one, we need information that explicitly describes, amongst others, the task performed by the unavailable service operation and the semantics of its input and outputs. This information is commonly encoded in the form of annotations that relates the service elements (i.e., operation, inputs and outputs) to concepts from ontologies that specify the semantics of the service elements in the real world. An ontology is commonly defined as an explicit specification of a conceptualisation [6]. Formally, an ontology θ can be defined as a set of concepts, $\theta = \{c1,\ldots,cn\}$. The concepts are related to each other using the sub-concept relationship, which links general concepts to more specific ones. For example, *ProteinSequence* is a sub-concept of *Sequence*, for which we write $ProteinSequence \sqsubseteq Sequence$. The concepts can also be connected by other binary relationships, depending on the specific semantics encoded by the ontology.

To characterise service replaceability we assume that the web services are annotated using the following ontologies.

Task ontology, θ_{task}*:* This ontology captures information about the action carried out by service operations within a domain of interest. In bioinformatics, for instance, an operation is annotated using a term that describes the *in silico*

analysis it performs. Example of bioinformatics analyses include *sequence alignment* and *protein identification*. To retrieve the task annotation of service operations we consider the function *task()* defined as follows: *task:* $OPS \rightarrow \theta_{task}$.

Resource ontology, $\theta_{resource}$*:* This ontology contains concepts that denotes public bioinformatics data sources used by analysis operation for performing their task. For example, the bioinformatics service operation *getUniprotEntry()* provided by the *DDBJ*[2] uses the *Uniprot*[3] database for fetching the protein entry having the accession number given as input. From the point of view of replaceability, it is sometimes important to specify the external data source used by the operation as well as its task. Two operations that perform the same task can deliver different outputs depending on the underlying data source they are using. Therefore, the user should be informed with this difference before taking the decision of whether to accept the located operation as a substitute of the unavailable one or not. To know the resource used by a given service operation we use the function *resource()*: *resource:* $OPS \rightarrow \theta_{resource}$

Domain ontology, θ_{domain}*:* This ontology captures information about the application domains covered by the operations, and enables us to describe the real world concepts to which each parameter corresponds. An example of such an ontology is that developed by the TAMBIS project [2] describing the domain of molecular biology. Examples of domain concepts include *Protein*, *DNA* and *RNA*. In this work, we assume the existence of a function *domain()* with the following signature: *domain:* $(INS \cup OUTS) \rightarrow \theta_{domain}$

3 Service Operation Replaceability

The semantic annotations of service operations presented in the previous section can be used to deal with the problem of service unavailability in workflows by supporting the user in the task of locating the service operations candidates for substituting the unavailable ones. We will use an example of a real *in silico* experiment that we have developed in ISPIDER, and *e*-Science project[4]. The experiment is used for performing value-added protein identification in which protein identification results are augmented with additional information about the proteins that are homologous to the identified protein [3]. Figure 1 illustrates the workflow that implements this experiment.

The workflow consists of three operations. The *IdentifyProtein* operation takes as input peptide masses obtained from the digestion of a protein together with an identification error and outputs the Uniprot accession number of the *"best"* match. Given a protein accession, the operation *GetHomologous* performs a homology search and returns the list of similar proteins. The accessions of the homologous proteins are then used to feed the execution of the *GetGOTerm*

[2] DNA Data Bank of Japan
[3] http://www.pir2.uniprot.org/
[4] http://www.ispider.man.ac.uk/

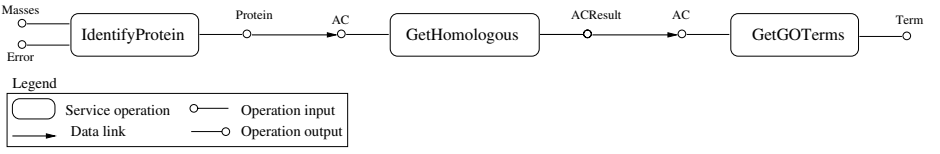

Fig. 1. Example workflow

operation to obtain their corresponding gene ontology term[5]. We have constructed this workflow two years before the time of writing. Recently, we received a request from a bioinformatician to use the workflow. However, because the operation *GetHomologous* that we used for performing the protein homology search does no longer exist, the user was unable to execute the workflow. Therefore, we had to search for an available web service that performs homology searches and that we can use instead. This operation turned out to be time consuming. We found several web services for performing homology searches and that are provided by the DNA Databank of Japan[6], the European Bioinformatics Institute[7] and the National Centre for Biotechnology Information[8]. Nonetheless, we had to try several service operations before locating an operation that can actually replace the *GetHomologous* operation within the protein identification workflow. The reason is that even though the service operations we found fulfil the task that the unavailable one does (i.e., protein homology search), they require and deliver parameters different from those that the unavailable operation has: some of the operations that we tried to use have input and output parameters that are mismatching with the input of *IdentifyProtein* operation and the input of the operation *GetGOTerm*. Moreover, some of the candidate operations were found to be using data sources that are different from that required by the unavailable operation and as a such were judged inappropriate. For example, we were unable to use the *Blastx* operation provided by the NCBI as it uses a nucleotide sequence data sources whereas the unavailable operation relies on a protein database for identifying similar proteins.

To support the user and facilitate the process of locating the service operations able to substitute the unavailable operations in a workflow (as illustrated in the example workflow presented above), we use the replaceability criteria presented in the following.

Task Replaceability. In order for an operation *op2* to be able to substitute an operation *op1*, *op2* must fulfil a task that is equivalent to or subsumes the task *op1* performs. Formally, *task(op1)* $\sqsubseteq$ *task(op2)*.

For instance, in the protein identification workflow illustrated in Figure 1, the unavailable operation *GetHomologous* performs a protein sequence alignment. An example of an operation that can replace *GetHomologous* in terms of task is the

[5] http://www.geneontology.org/
[6] http://www.ddbj.nig.ac.jp/
[7] http://www.ebi.ac.uk/
[8] http://www.ncbi.nlm.nih.gov/

operation *SearchSimple* provided by the DDBJ and which aligns bioinformatics sequences: *ProteinSequenceAlignment* $\sqsubseteq$ *BioinformaticsSequenceAlignment*.

Resource Replaceability. If an operation uses an external data source for performing its task, then the operation that replaces it within the workflow must uses the same data source. Formally, an operation *op2* can replace an operation *op1* in terms of resource iff *resource(op1)* = *resource(op2)*

Resource replaceability can be of extreme importance when locating substitutes for a service operation in scientific workflows. Indeed, workflows are being recognised as a mean for validating the results claimed by their authors (scientists) and which may, for instance, be demonstrating a certain scientific fact. To verify the claims of the authors of the workflow, other scientists execute the same workflow and examine the execution results to see whether it is compatible with the authors' conclusions or not. Therefore, it is important that the workflow execution reproduces the same results as those obtained by the authors of the workflow in the first instance. For the workflow execution to reproduce the same results, the substitute operations in the workflow should use the same resource as their counterpart unavailable operations.

Parameter Compatibility. When substituting an operation *op1* in a workflow *wf* with another operation *op2*, new data links must be defined to connect the input and output parameters of *op2* to operation parameters that were previously connected to *op1*'s parameters within the workflow *wf*. These data links are established by using parameter compatibility rules that ensure that a defined data link connects an output parameter to an input able to consume the data produced by that output. We adopt two parameter compatibility rules that we used in a previous work to identify mismatches in workflows [4].

Data type. Two connected output and input parameter are compatible in terms of data type iff the data type of the output is the same as or a subtype of the data type required by the input parameter. Formally, the output *(op,o)* is compatible with the input *(op',i)* in terms of data type iff[9]: *type(op,o)* $\preceq$ *type(op',i)*

Domain compatibility. In order to be compatible, the domain of the output must be equivalent to or a sub-concept of the domain of the subsequent input. Formally, the output parameter *(op,o)* is domain compatible with the input parameter *(op',i)* iff: *domain(op,o)* $\sqsubseteq$ *domain(op',i)*

The parameter compatibility criteria just presented are used to draw the correspondences between the parameters of the operation *op2*, a candidate for substituting an operation *op1* in a workflow *wf*, and the operations parameters previously connected to *op1* in *wf*. If for every mandatory input of *op2*, there exists a corresponding operation output in the workflow *wf*, and for every output parameter that were previously connected to the unavailable operation *op1*, there exists a corresponding operation input in the workflow *wf*, then the operation *op2* is added to the list of substitutes of *op1* in the workflow *wf*.

[9] The symbol $\preceq$ stands for subtype of.

4 Conclusions

The volatility of web services is one of the main issues that hinder the sharing and the reuse of scientific workflows. To our knowledge, there does not exist any work that attempted to address this issue. However, there are some proposals that address problems that are closely related. For example, Benatallah *et al* presents replaceability as a means for adapting a web service to different applications [5]. The problem they tackle is, however, different from ours: they focus on the replaceability of a web service from the business protocol perspective (i.e., the series of interactions with the service operations). Akram *et al* defined a framework for detecting changes in web services and reacting to those changes by redirecting the requests to the unavailable service to another service [1]. Using this framework, however, the discovery query used for locating the substituting service is manually specified by the service user. The method presented here compile the aspects considered by these proposals for characterising service operations and considers additional characteristics that are peculiar to the analysis operations that compose scientific workflows, e.g., the data source used by the service operation for performing the analysis.

References

1. Akram, M.S., Medjahed, B., Bouguettaya, A.: Supporting dynamic changes in web service environments. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 319–334. Springer, Heidelberg (2003)
2. Baker, P.G., Goble, C.A., Bechhofer, S., Paton, N.W., Stevens, R., Brass, A.: An ontology for bioinformatics applications. Bioinformatics 15(6), 510–520 (1999)
3. Belhajjame, K., Embury, S.M., Fan, H., Goble, C.A., Hermjakob, H., Hubbard, S.J., Jones, D., Jones, P., Martin, N., Oliver, S., Orengo, C., Paton, N.W., Poulovassilis, A., Siepen, J., Stevens, R., Taylor, C., Vinod, N., Zamboulis, L., Zhu, W.: Proteome data integration: Characteristics and challenges. In: UK All Hands Meeting (2005)
4. Belhajjame, K., Embury, S.M., Paton, N.W.: On characterising and identifying mismatches in scientific workflows. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 240–247. Springer, Heidelberg (2006)
5. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 415–429. Springer, Heidelberg (2005)
6. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220 (1993)