Fostering Scientific Workflow Preservation Through Discovery of Substitute Services

Khalid Belhajjame, Carole Goble, Stian Soiland-Reyes
School of Computer Science,
University of Manchester
Oxford Road, Manchester, UK
Email: first_name.last_name@manchester.ac.uk

David De Roure
Oxford e-Research Centre
University of Oxford,
Oxford, UK
Email: david.deroure@oerc.ox.ac.uk

***Abstract*—Scientific workflows are increasingly gaining momentum as the new paradigm for modeling and enacting scientific experiments. The value of a workflow specification does not end once it is enacted. Indeed, workflow specifications encapsulate knowledge that documents scientific experiments, and are, therefore, worth preserving.**

Our experience suggests that workflow preservation is frequently hampered by the volatility of the constituent service operations when these operations are supplied by third-party providers. To deal with this issue, we propose a heuristic for locating substitutes that are able to replace unavailable service operations within workflows. The proposed method uses the data links connecting inputs and outputs of service operations in existing workflow specifications to locate operations with parameters compatible with those of the missing operations. Furthermore, it exploits provenance traces collected from past executions of workflows to ensure that candidate substitutes perform tasks similar to those of the missing operations. The effectiveness of the proposed method has been empirically assessed.

***Index Terms*—scientific workflow, workflow preservation, workflow provenance, web services**

I. Introduction

The wide adoption of scientific workflows, as a mechanism for loosely aggregating existing services, has dramatically revolutionized the way many scientists conduct their daily experiments as suggested by the increasing number of scientific disciplines that have embraced workflow technology, e.g., bioinformatics, biomedical informatics, cheminformatics, ecoinformatics and geoinformatics. 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.

The value of a workflow specification does not end once it is enacted. Indeed, workflow specifications encapsulate knowledge that documents scientific experiments, and are, therefore, worth preserving. Such specifications can be used, for example, by other scientists as building blocks for designing new experiments, or used by reviewers to assess the reproducibility of the results claimed by the workflow authors.

A step in this direction, we aim in the context of the Wf4Ever project [1], to bring new functionality to scientists, enabling them to take a step forward in the preservation of scientific knowledge by introducing the novel concept of workflow-related Research Objects, which acknowledges the central role of workflows in e-science and their relevance for scientific preservation. We address the preservation requirements of scientific data by considering workflows as live entities, which as they evolve need to be kept consistent with respect to research materials, many of them beyond the control of the originating scientists.

In a context where the service operations that constitute the workflow are provided by the institution that wishes to preserve the worklow, workflow preservation can be (partially) guaranteed by ensuring that the constituent services are continuously supplied. The situation is different when the services are supplied by third party providers. In fact, a problem that frequently hampers the preservation of such workflows is the volatility of the services that constitute the workflows. This is not surprising; there is generally no agreement between service providers and users that compel the providers to continuously supply their services.

We have shown in a previous study [3] that semantic annotations of web services can be used to identify suitable substitutes for unavailable service operations, thereby enabling the preservation of scientific workflows. Specifically, we developed an algorithm that, given as input annotations that semantically describe the missing service operations using concepts from domain ontologies [22], [13], identifies available service operations that can play the same role as the unavailable ones.

While, the algorithm we developed is sound, its practical applicability is hindered by the following facts. First, semantic annotations of web services are scarce: the number of web services that are annotated lags well behind the growing number of available web services. As a result, the likelihood of locating substitute operations using the algorithm described in [3], is small. Second, our experience with QuASAR[2], a tool that we developed for testing semantic web service annotations, suggests that a large proportion of existing semantic annotations suffer from inaccuracies: annotators tend to use concepts that are subconcepts or superconcepts of the concepts that should be used for annotating web service

[1]http://www.wf4ever-project.org

[2]http://img.cs.man.ac.uk/quasar/verification.php

parameters. As a result, a substitute that is discovered for replacing an unavailable operation using such annotations may turn out to be unsuitable, and, inversely, a suitable substitute may be discarded. Finally, scientific workflows may contain operations that are implemented using mechanisms other than web services, e.g., local programs or scripts, the annotations of which are not available.

To deal with the above issues, we propose in this paper a heuristic for locating substitutes in the absence of semantic annotations of web services. The proposed method uses the data links connecting inputs and outputs of service operations in existing workflow specifications to locate operations with parameters compatible to the missing operations. Furthermore, it exploits provenance data [17] collected from past executions of workflows to ensure that candidate substitutes perform tasks similar to those of the missing operations.

The paper is organised as follows. We begin (in Section II) by formally defining scientific workflows. We then show how substitutes can be located in the absence of semantic annotations (in Section III). To assess the value of the proposed approach in practice, we report on the result of a preliminary evaluation (in Section IV). We analyse and compare our work to existing proposals in the literature in Section V, and conclude the paper in Section VI underlining our main contributions.

II. Data-Driven Workflows

The method we propose for locating substitutes for unavailable service operations takes workflow specifications as input. These can be defined in conventional workflow languages in which dependencies between the constituent operations are defined in terms of both dataflow and controlflow, e.g., BPEL [19]. However, given that the method we present here does not exploit controlflow dependencies between operations, we focus in the rest of the paper on data driven workflows. We define a data driven 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*.

Operation: An operation represents the unit of functionality supplied by a service. It is defined as:

$$op = \langle nameOp, loc, desc \rangle,$$

where *nameOp* is the operation identifier, it must be unique within the service, *loc* is the end point of the service to which the operation belongs (it also serves as the service identifier), and *desc* is a textual description of the action performed by the operation.

Parameter: 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). In the following we will use *inputs(op)* and *outputs(op)* to denote the input parameters and the output parameters of a service operation *op*, respectively.

Input parameters can be either mandatory or optional, we assume the existence of the function *mandatory(op)* that returns the set of mandatory input parameters of the operation *op*.

Data links: 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, i.e.:

$$IN = \bigcup_{op\ \in\ wf.OP} inputs(op)$$
$$OUT = \bigcup_{op\ \in\ wf.OP} outputs(op)$$

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 the following notations:

- *WFS* is the domain of workflows.
- *OPS* is the domain of service operations.
- *DLS* is the set of all data link connections in *WFS*, i.e., $DLS = \{\ dl \mid dl \in DL\ \wedge\ \langle \_, \_, DL \rangle \in WFS \}$.
- *INS* is the set of all the inputs of the operations in *OPS*, i.e., $INS = \bigcup_{op\ \in\ OPS} inputs(op)$.
- *OUTS* is the set of all the outputs of the operations in *OPS*, i.e., $OUTS = \bigcup_{op\ \in\ OPS} outputs(op)$.
- *connectedParams(op,wf)* returns the set of parameters that are connected to the input or output parameters of the operation *op* within the workflow *wf*.

III. Discovering Substitutes For Volatile Service Operations

As described in the introduction, because of the scarcity of semantic annotations of web services the likelihood of locating substitute operations using semantic annotations is small. This raises the question as to *how a substitute can be located when semantic annotations of web services are not available?*

A solution that we explore in the rest of this paper uses as input existing workflow specifications to locate substitutes. To illustrate this idea, we will use an example of a real *in silico* experiment that we have developed in ISPIDER, an *e*-Science project[3]. 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 [1]. Figure 1 illustrates the workflow that implements this experiment.

[3]http://www.ispider.man.ac.uk/

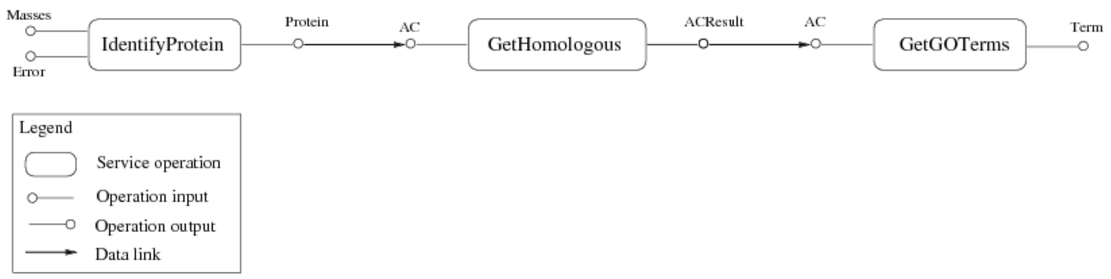

Fig. 1. Value-added protein identification workflow (identified in the text by $proteinIdentificationWf_1$)

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* operation to obtain their corresponding gene ontology term[4]. We constructed this workflow four years before the time of writing. Later, we received a request from a bioinformatician from the ^{my}Grid project[5] to use the workflow. However, because the operation *GetHomologous* that we used for performing the protein homology search did 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 Center 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 fulfill 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 output of the *IdentifyProtein* operation and the input of the *GetGOTerm* operation.

The rest of this section presents the elements of the solution that we propose for locating suitable substitutes for unavailable service operations.

A. Parameter Compatibility

Consider the existence of the workflow illustrated in Figure 2. The operation *GetSimilarProteins* in this workflow is connected to both *IdentifyProtein* and *GetGOTerm* which are also connected to the unavailable operation *GetHomologous*. If the data links in this workflow are free from mismatches then the input and the output of *GetSimilarProtein* are respectively compatible with the output of *IdentifyProtein* and the input of *GetGOTerm* in that they can be connected within a workflow.

Formally, let *wf1* be a workflow in which the operation *op1* is unavailable. The operation *op2* can replace the operation *op1* in terms of its inputs and outputs if for every parameter *(op',p')* that is connected to *op1* in the workflow *wf1*, there exists a workflow *wf2* in *WFS* in which the parameter *(op',p')* is connected to *op2*. That is:

$$\forall\,(op', p') \;\in\; connectedParams(op1, wf1),$$
$$\exists\; wf2 \;\in\; WFS,$$
$$(op', p') \;\in\; connectedParams(op2, wf2)$$

For example, the input and output parameters of the operations connected to the *GetHomologous* operation within the workflow $proteinIdentificationWf_1$ illustrated in Figure 1 are connected to the *GetSimilarProteins* operation within the workflow $proteinIdentificationWf_2$ illustrated in Figure 2. Therefore, the operation *GetSimilarProteins* can replace the operation *GetHomologous* in terms of inputs and outputs within the workflow in $proteinIdentificationWf_1$. In the following, we consider the existence of the function $compatibleParams(op)$, that, given an operation *op*, returns the set of operations that can replace *op* in terms of input and output parameters.

Notice that in addition to the condition stated above, all the mandatory input parameters of the substitute must be connected within the workflow in which it is used. For example, if the operation *GetSimilarProteins* had a mandatory input other than $(GetSimilarProteins, i)$, then it would have not been possible to use it as a replacement for the *GetHomologous* operation.

In the following we assume the existence of the function *map()* with the following signature:

$$map : (INS \,\cup\, OUTS) \,\times\, OP \,\times\, WFS \,\rightarrow\, (INS \,\cup\, OUTS)$$

Given a parameter $(op1, p1)$ and a workflow *wf*, $map((op1, p1), op2, wf)$ returns the parameter $(op2, p2)$ that can play the same role as $(op1, p1)$ within the workflow *wf*. For example:

$$map((GetHomologous, AC), GetSimilarProteins,$$
$$proteinIdentificationWf_1) \;=\; (GetSimilarProteins, i)$$
$$map((GetHomologous, ACResult), GetSimilarProteins,$$

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

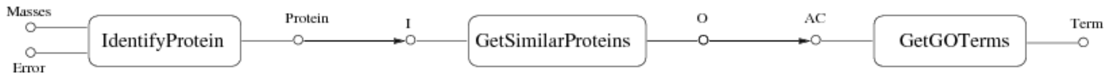

Fig. 2. Value-added protein identification workflow (identified in the text by $proteinIdentificationWf_2$)

$$proteinIdentificationWf_1) \;=\; (GetSimilarProteins, o)$$

map() is a partial function in the sense that an operation may not have a parameter that can play the same role as the parameter of a missing operation within a given workflow.

B. Task Compatibility

In addition to the compatibility in terms of inputs and outputs, we have to check that the candidate substitute performs a task compatible with that of the unavailable operation. This raises the question as to *how we can verify that a candidate substitute performs a task that is compatible with that of the unavailable operation in the absence of semantic web service annotations*.

To perform this test, we exploit the following observation. If *GetSimilarProteins* performs a task compatible with that of *GetHomologous*, then for the same input, both operations should in principle deliver the same output. Formally, an operation *op2* is able to replace the operation *op1* in terms of task, if for every possible input instances that *op1* is able to consume, *op2* delivers the same output as that obtained by invoking *op1*. To perform this test, however, we will have to call the missing operation *op1*!

A solution that we adopt for overcoming the above problem makes use of workflow provenance logs. These are traces that contain information about past executions of workflows. More importantly, they contain intermediate data that were used as input and delivered as output by the constituent operations of a workflow when enacted. Example of workflow provenance stores are Janus [15] and PASOA [14]. The rest of this section shows the method we use to verify that a given operation performs a task compatible with that of the missing operation using as input collected workflow provenance logs.

Given a candidate operation that was located based on parameters connections in workflows as illustrated earlier, we invoke it using the same inputs that were used in the past to invoke the unavailable operation and which are retrieved from provenance logs. We then compare the results of unavailable and candidate substitute. The candidate is judged to be a suitable substitute if it delivers the same results as those obtained in the past by invoking the unavailable operation. To formally express this test we define an operation execution by the triple:

$$\langle op,\; I,\; O\rangle$$

I is a set of pairs $((op, i), ins_i)$ where (op, i) is an input parameter of *op* and ins_i is the instance used to feed (op, i). Similarly, O is a set of pairs $\langle (op, o), ins_o\rangle$ where (op, o) is an output parameter of *op* and ins_o is the value of (op, o) that was obtained from the execution of *op* with the input values given in I_{inputs}.

To retrieve operation executions, we use the function *getExecutions*, the signature of which is illustrated below.

$$getExecutions : OP \;\rightarrow\; \mathcal{E}$$

where $\mathcal{E}$ is the domain of operation executions. We also assume the existence of the function *value()* to retrieve the value associated with an input/output within an execution.

$$value : (INS \;\cup\; OUTS) \;\times\; \mathcal{E} \;\rightarrow\; \mathcal{V}$$

where $\mathcal{V}$ is the domain of values of input and output parameters.

Let *op1* be an unavailable operation and *op2* be an operation with inputs and outputs compatible with those of *op1*. *op2* may be compatible in terms of task with *op1* if:

$$\begin{array}{l}\forall\, e1 \;\in\; getExecutions(op1),\; \exists\, e2 \;\in\; getExecutions(op2),\\ such\ that:\\ \quad \forall (op1, i1) \;\in\; inputs(op1),\\ \quad value((op1, i1), e1) \;=\; value(map((op1, i1), op2, wf1), e2)\\ \quad and\\ \quad \forall (op1, o1) \;\in\; outputs(op1),\\ \quad value((op1, o1), e1) \;=\; value(map((op1, o1), op2, wf1), e2)\end{array}$$

Notice that we say *may be compatible*. This is because we may not be able to compare the outputs obtained for every possible input value of the operation *op1*. Provenance logs is likely to provide only a subset of possible input values that the unavailable operation was able to consume and, therefore, may not include values for which the candidate operation delivers results that are different from those obtained using the unavailable operation.

C. Relaxing Substitutability Conditions

The above conditions for checking the suitability of an operation as a substitute for another one may be stronger than is required in practice. The bioinformatics field in particular, the input and output parameters of bioinformatics analysis operations are weakly typed. Most of the time, they are typed as Strings no matter how complex are their contents. More importantly, input and output parameters of the same semantic domain can be formatted using a multitude of representations. There are various text representations that are adopted in bioinformatics. For example, a biological sequence can be formatted according to Fasta, Ensemble, Uniprot, gff formats to cite a few. Because of representation mismatch, a service operation that has input (resp. output) parameters that belong to the same semantic domain as those of the missing operation, and that performs the same task as the missing operation, may be found not to be a suitable substitute. To overcome such a problem, we relax the above condition as illustrated below by

```
>sp|P17110|CH36_CERCA Chorion protein S36 OS=Ceratitis
MNCFLFTLFFVAAPLATASYGSSSGGGGGGSSYLSSASSNGLDELVQ
AAAGGAQQAGGTITPANAEIPVSPAEVARLNQVQAQLQALNSNPV
YRNLKNSDAIAESLAESSLASKIRQGNINIVAPNVIDQGVYRSLLVPS
GQNNHQVIATQPLPPIIVNQPALPPTQIGGGPAAVVKAAPVIYKIKP
SVIYQQEVINKVPTPLSLNPVYVKVYKPGKKIDAPLVPGVQQNYQA
PSYGGSSYSAPAASYEPAPAPSYSAAPAQSYNAAPAPSYSAAPAASY
GAAPSASYDAAPAASYGAESSYGSPQSSSSYGSAPPASGY
```

Fig. 3. A protein entry formatted using the Fasta representation

requiring that the output values of unavailable and substitute to be similar (as opposed to equal).

$\forall\, e1 \in getExecutions(op1),\ \exists\, e2 \in getExecutions(op2),$
$such\ that$:
$\forall (op1, i1) \in inputs(op1),$
$value((op1, i1), e1) = value(map((op1, i1), op2, wf1), e2)$
and
$\forall (op1, o1) \in outputs(op1),$
$sim(value((op1, o1), e1), value(map((op1, o1), op2, wf1), e2))$
$\geq w$

sim() is a string similarity function [23] that returns a double ranging from 0 to 1 specifying how similar the values given as input. It returns 1 when the input values of the function are equal. *w* is a value between 0 and 1 that can be specified by the user; it represents the level of similarity required. Examples of similarity metrics that can be used for this purpose as the Euclidian and Cosine metrics [23].

The use of similarity metrics can be helpful in identifying identical entries when the difference between representations is minor. When the difference in representation is large, two entries that are identical may be found to be different even when similarity metrics are employed to compare parameters values. To illustrate this, consider the protein sequences that are formatted according to Fasta and Uniprot formats and which are illustrated in Figure 3 an Figure 4, respectively. These protein sequences designate the same protein, yet if we simply compare their content, they are different. The use of similarity metrics to compare the two entries does not help. For example, the similarity score obtained using the Cosine similarity is 0.07, which is a low similarity score. This is partly due to the fact that when comparing (complex) parameter instances, the above solution treats them as textual content; in other words, it does not distinguish representation from actual content.

To overcome the above problem, we use a two-step approach in which given a parameter value, we first try to derive its representation. This step is performed using a recognizer that is able to recognize widely popular biological formats. Some representations contain key attributes that can be used to uniquely identify a biological entry that is represented by the parameter instance. This is the case for example for biological sequence records. If that is the case, the recognizer extracts the value of such an attribute. For example, the attribute used to identify entries formatted using the Ensembl format is AC, where as in Fasta format, the identifier is specified in the first line using the regular expression $sp|identifier|$.

```
ID   CH36_CERCA          Reviewed;         320 AA.
AC   P17110;
DT   01-AUG-1990, integrated into
DT   UniProtKB/Swiss-Prot.
DT   01-AUG-1990, sequence version 1.
DT   05-OCT-2010, entry version 44.
DE   RecName: Full=Chorion protein S36;
DE   Flags: Precursor;
GN   Name=Cp36; Synonyms=S36;
OS   Ceratitis capitata (Mediterranean fruit fly)
OC   Eukaryota; Metazoa; Arthropoda; Hexapoda;
OC   Neoptera; Endopterygota; Diptera; Brachycera;
OC   Tephritoidea; Tephritidae; Ceratitis; Ceratitis.
OX   NCBI_TaxID=7213;
KW   Repeat; Secreted; Signal.
SQ   SEQUENCE   320 AA;  32319 MW;  ECF9B72FFBE54C61
MNCFLFTLFFVAAPLATASYGSSSGGGGGGSSYLSSASSNGLDELVQ
AAAGGAQQAGGTITPANAEIPVSPAEVARLNQVQAQLQALNSNPV
YRNLKNSDAIAESLAESSLASKIRQGNINIVAPNVIDQGVYRSLLVPS
GQNNHQVIATQPLPPIIVNQPALPPTQIGGGPAAVVKAAPVIYKIKP
SVIYQQEVINKVPTPLSLNPVYVKVYKPGKKIDAPLVPGVQQNYQA
PSYGGSSYSAPAASYEPAPAPSYSAAPAQSYNAAPAPSYSAAPAASY
GAAPSASYDAAPAASYGAESSYGSPQSSSSYGSAPPASGY
//
```

Fig. 4. A protein entry formatted using the Ensembl representation

Now to compare two entries such as those presented earlier, if such entries are associated with identifiers, then rather than comparing their content, we compare their identifiers. Using this approach, the above two entries are found to be identical: both are identified using the term $P17110$.

In the following we consider the existence of the boolean function $isTaskCompatible(op, op')$, which is true if the operation op' performs the same task as the operation op, and is false, otherwise.

Now that we have presented all elements of the solution, we can proceed to the description of the algorithm used to locate substitutes.

The algorithm is listed in Figure 5. The algorithm is composed of two main steps. First, given a missing operation op_m, the set of operations $Candidate\_op_s$ that are able to replace op_m in terms of input and output parameters are located *(line 2)*. The operations in $Candidate\_op_s$ that perform the same task as op_m are then added to the set of suitable substitutes *(lines 3-5)*.

```
Algorithm LocateSubstitutes
Inputs  $op_m$ : an unavailable operation
Outputs $OP_s$ : A set of substitutes for $op_m$
Begin
1  $OP_s \leftarrow \emptyset$
2  $Candidate\_OP_s \Leftarrow compatibleParams(op_m)$
3  For each $op$ in $Candidate\_OP_s$ Do
4          If $isTaskCompatible(op_m, op)$
5          Then $OP_s \leftarrow OP_s \cup \{op\}$
6  Return $OP_s$
End
```

Fig. 5. Algorithm used for locating substitutes.

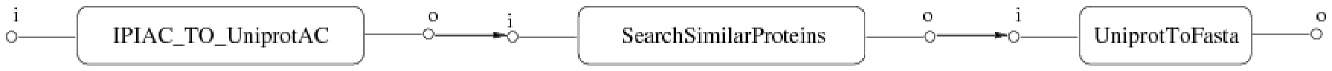

Fig. 6. Example of a workflow that can be used to replace the service operation *GetHomologous* in the protein identification workflow

D. Replacing Missing Operation with a Workflow

As mentioned earlier, in certain cases, we may locate substitutes operations that, although, performs the same task and take inputs and outputs that are compatible with the missing operations in that they belong the same semantic domain, the input and output parameters of the substitute may be formatted using representations that are different from those used by the missing operations. To resolve representation mismatches, we use a method in which the candidate substitute is combined within a workflow to other service operations that transform the representation of its inputs and outputs as required. The obtained workflow is then used to substitute the unavailable operation. This is perhaps better explained using a concrete example. In the protein identification workflow, we found a service operation, *SearchSimilarProteins* that performs the same task. As far as the input and output parameters are concerned, *SearchSimilarProteins* has an input that is compatible with the output of the operation *IdentifyProtein* in terms of data type and semantic domain, however, because *IdentifyProtein* outputs protein accession that is formatted according to the *IPIAccession* format and *SearchSimilarProteins*'s input requires the protein accession to be formatted using the *UniprotAccession* representation, the two parameters cannot be connected. Similarly, *SearchSimilarProteins* outputs protein records formatted using the *Fasta* representation whereas *GetGOTerm*'s input requires protein records formatted according to the *UniprotRecord* format and, thus, the two parameters cannot be connected. To resolve this mismatch in representation, Figure 6 shows that the operation *SearchSimialrProteins* is combined with two operations *IPIAC_TO_UniprotAC* and *UniprotToFasta*. The former is used to transform the IPI protein accession delivered by *IdentifyProtein* into Uniprot accession, whereas the second is used for transforming Uniprot records into Fasta records. Both *IPIAC_TO_UniprotAC* and *UniprotToFasta* are mapping operations that do not alter the semantic domain of the input and output of the operation *SearchSimialrProteins*. They are domain preserving mappings: their inputs and outputs belong to the same semantic domain, e.g., *IPIAC_TO_UniprotAC* consumes a protein accession and outputs a protein accession. These kind of service operations are commonly used in scientific workflows to deal with representation mismatch [4], [5] and are known in the literature of scientific workflows as *Shims* [20], [10]. The resulting workflow can be used to substitute the operation *GetHomologous* in the protein identification workflow.

IV. Preliminary Evaluation

To assess and gain some insight on how effective is the method presented in this paper, we conducted an experimental evaluation using as input real world scientific workflows. Specifically, we used 10 workflows that were developed in the context of *e*-science projects such as ^{my}Grid, ISPIDER[9] and Embrace[10]. We chose workflows for which example input parameters are available to be able to log provenance traces for their executions. To collect data provenance for these workflows, we enacted them using sample of inputs and logged the execution traces using Janus [16], the provenance system of the Taverna workbench [18].

We then created for each workflow, an identical (clone) workflow, and change a given service operation in the clone workflow with a *fictive* operation that does not exist. For example, Figure 7 shows a workflow that, given a biological accession, aligns the corresponding biological sequence and plots the alignment report in a human readable form. We created a clone workflow in which the operation *searchSimple* was replaced by the operation *blastSimple*, which does not exist, as shown in Figure 8.

Given the provenance we collected earlier for workflows, we made some changes to associate the instances of the inputs and outputs of the operations replaced with those of the corresponding *fictive* operations. For example, we associated the input and output instances of the *searchSimple* operation used in the workflow shown in Figure 7 with the input and output of the *blastSimple* operation used in the workflow shown in Figure 8.

Given the workflows obtained by the above operation, we then run our algorithm for identifying substitutes for the *fictive* operations that do not exist. Ideally, each *fictive* operation should be replaced by the corresponding operation in the initial workflow. For example, the substitute for the operation *blastSimple* is the operation *searchSimple*, or an operation that has parameters that are compatible with those of *searchSimple* and performs the same task as that performed by *searchSimple*.

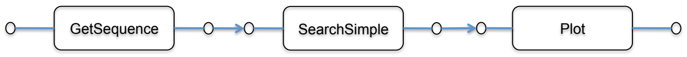

Fig. 7. The above workflow performs a sequence alignment, and plots the results in a form that is human readable.

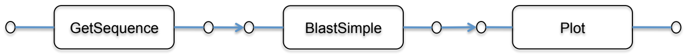

Fig. 8. The above workflow is a clone of the workflow illustrated in Figure 7, in which the operation *searchSimple* was replaced with the operation *blastSimple*, which does not exist.

Using the algorithm presented in this paper for locating substitutes, we were able to locate suitable substitutes for 9

[9]http://www.ispider.manchester.ac.uk/cgi-bin/ispider.pl
[10]http://www.embracegrid.info

service operations out of 10. For the remaining operation, the algorithm identified 2 operations, one suitable, which is *getSwissEntry*, and one unsuitable, *getDDBJEntry*. After analysis, it transpired that the provenance collected for these two operations were erroneous. Both operations were associated with the same input and output instance. The input is an empty string, and the output is an error message stating that the input is not valid, specifically, the output is the following string *'there is no value in the parameter accession'*.

To see whether the method we proposed to deal with heterogeneity in representation is effective, we labeled the operation *KeggMapper* within one of the workflows as unavailable. This operation takes as input a gene entry that is formatted according to the *EMBL* representation and retrieves the corresponding biological pathways. We labeled this operation as missing, and insert the operation *Bconv* supplied by the *Kegg* project[11] as a candidate service operation. The operation *Bconv* performs the same task and takes input and output parameters that are semantically compatible with those of *KeggMapper*. However, the *Bconv* operation requires gene entries that are formatted using the *Genbank* format. Using our method, the operation *Bconv* was identified as suitable substitute even in the presence of the representation mismatch. This is thanks to the use of the recognizer which identified two input instances of the operations *KeggMapper* and *Bconv* as identical, i.e., they represents the same gene even if they are formatted using different representations. To resolve this representation mismatch, we used our tool [2] to locate a mapping operation that transforms *EMBL* identifiers into *Genbank* identifiers, *EMBLToGenbank*. The substitute of the operation *KeggMapper* was then a workflow obtained by composing in sequence *EMBLToGenbank* to *Bconv* (see Figure 9).

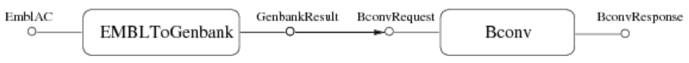

Fig. 9. The workflow constructed for substituting the operation *KegMapper*

In summary, the results of this small study showed that locating substitutes in the absence of semantic annotations is possible if provenance logs of unavailable operations are available. The experiment also showed that error in provenance logs due to the fact that operation input and output instances can be error messages, instead of legal values, may pose problems and lead to the discovery of unsuitable substitutes.

While the above evaluation results are encouraging, further larger scale experiments that involve thousands of service operations are needed. Locating substitutes when the number of services is large can be expensive as every service may need to be executed once or more. These are issues that we are looking at as part of our ongoing research investigations.

V. Related Work

While document preservation is a long standing research theme [21], computation (and therefore workflow) preservation is a relatively new topic, which has been gaining interest in the last few years. For example, the ACM SIGMOD repeatability effort which has been organized twice since 2008 set up as a goal to verify that the experiments published in the paper accepted at the conference are repeatable [12]. Also, Elsevier, a publisher of scientific, technical and medical information products and services, has recently organized the Executable Paper Grand Challenge[12], to address the difficulties associated with reproducing computer science research results. As a result, few systems have emerged for capturing computation in the form of workflows that can be shared and preserved. For example, Koop *et al.* [11] have developed an infrastructure for capturing computations and experiments in the form of workflows that are managed using the VisTrails workflow systems. Frew *et al.* [9] developed a system for collecting provenance that enable the specification of experiments and computations in the form of workflows.

Our work is complementary to these outlined efforts: while the above proposals aim to capture computations and experiments in the form of workflows that can be stored and shared, our work aims at ensuring the preservation of such workflows by supporting the task of locating substitutes for volatile services.

Our work is also related to proposals that aim to support service matching. The issue addressed by such proposals can be formulated as the problem of locating web services that match a user request [7]. For example, Woogle [8] is a search engine for web services. It is used for finding similar service operations or operations that compose with one another. The matching algorithm implemented by woogle utilizes clustering techniques to group the inputs/outputs of service operations into groups of parameters. Based on the assumption that similar parameters belong to the same group of parameters, two operations are considered to match if their respective inputs/outputs belong to the same group of parameters.

Cardoso *el al* [6] proposed a method for constructing workflows using which the constituent services of the workflow are located by matching service templates that specify the characteristics of the services to be incorporated within the workflow, against the available services. Like Woogle, they rely on similarity measures for locating the services that match the desired service templates. Specifically, they propose algorithms that measure the similarity of the task, inputs and outputs of the available services to those specified in the service templates.

In the same lines, Radetzki *et al.* [20] developed a method for discovering *shims*, which are service operations the are used for resolving mismatches within scientific workflows. In doing so, they rely on similarity techniques to infer semantic information about web services, and use the derived semantic annotations to match web services.

The method presented in this paper differs from the above proposals in the sense that it does not rely on semantic annotations of services, which are scarce, or on similarity

[11]http://www.genome.jp/kegg/

[12]http://www.executablepapers.com

metrics that match service operations by comparing their names and the names of their parameters, which can be misleading, but rather uses examples of data instances that are consumed and produced by service operations and that are captured in the form of provenance to compare missing operation and candidate substitutes.

VI. Conclusions

The volatility of web services is an issue that frequently arises in practice when sharing, reusing and preserving scientific workflows that use third-party web services. We proposed in this paper a solution that can assist designers in curating their workflows by locating substitute operations able to replace unavailable ones.

The solution we propose has been evaluated against a small number of real world scientific workflows. As mentioned in the evaluation section, larger scale experiments are needed in order to assess the performance of our method. As part of our ongoing work, we are investigating the limitations in terms of performance that the method presented may suffer from and design heuristics that can be used to improve the efficiency of the method, e.g., by avoiding the greedy search we are adopting at present for locating substitutes and opt for more sophisticated search strategies. Also, we intend to investigate the capabilities of existing instance-based matching algorithms in identifying compatible operation parameters. Notice that the solution proposed relies on the availability of repositories that store workflow provenance traces. While currently such repositories are far from being wide-spread, we anticipate an increase in their number as a result of the efforts being made by the scientific workflow community to standardize, publish and share workflow provenance[13].

As part of our future work, we are investigating how our solution can benefit workflow users in practice, e.g., the users within the life science and astronomy communities. In particular, we are examining its use within the context of the Wf4Ever project, a European project that aims to provides the means for ensuring the preservation of scientific workflows and myExperiment[14], a project that is developing a publicly accessible repository and associated tools to allow scientists to publish, annotate and discover workflows.

References

[1] K. Belhajjame, S. M. Embury, H. Fan, C. A. Goble, H. Hermjakob, S. J. Hubbard, D. Jones, P. Jones, N. Martin, S. Oliver, C. Orengo, N. W. Paton, A. Poulovassilis, J. Siepen, R. Stevens, C. Taylor, N. Vinod, L. Zamboulis, and W. Zhu. Proteome data integration: Characteristics and challenges. In *UK All Hands Meeting*, 2005.

[2] K. Belhajjame, S. M. Embury, and N. W. Paton. On characterising and identifying mismatches in scientific workflows. In *DILS*, pages 240–247. Springer, 2006.

[3] Khalid Belhajjame. Semantic replaceability of escience web services. In *eScience*, pages 449–456. IEEE Computer Society, 2007.

[4] Khalid Belhajjame, Suzanne M. Embury, and Norman W. Paton. On characterising and identifying mismatches in scientific workflows. In *3rd International Workshop on Data Integration in the Life Sciences (DILS 06)*, pages 240–247. Springer, 2006.

[5] Shawn Bowers and Bertram Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In Erhard Rahm, editor, *DILS*, volume 2994 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[6] Jorge Cardoso and Amit P. Sheth. Semantic e-workflow composition. *J. Intell. Inf. Syst.*, 21(3):191–225, 2003.

[7] Jorge Cardoso and Amit P. Sheth, editors. *Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004, Revised Selected Papers*, volume 3387 of *Lecture Notes in Computer Science*. Springer, 2004.

[8] Xin Dong, Alon Y. Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Simlarity search for web services. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 372–383. Morgan Kaufmann, 2004.

[9] J. Frew, G. Janée, and P. Slaughter. Automatic provenance collection and publishing in a science data production environment - early results. In *IPAW*, pages 27–33. Springer, 2010.

[10] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole A. Goble, Matthew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web-Server-Issue):729–732, 2006.

[11] D. Koop, E. Santos, P. Mates, H. T. Vo, P. Bonnet, B. Bauer, B. Surer, M. Troyer, D. N. Williams, J. E. Tohline, J. Freire, and C. T. Silva. A provenance-based infrastructure to support the life cycle of executable papers. *Procedia CS*, 4:648–657, 2011.

[12] I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, S. Zoupanos, and D. Shasha. The repeatability experiment of sigmod 2008. *SIGMOD Record*, 37(1):39–45, 2008.

[13] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[14] Simon Miles, Sylvia C. Wong, Weijian Fang, Paul T. Groth, Klaus-Peter Zauner, and Luc Moreau. Provenance-based validation of e-science experiments. *J. Web Sem.*, 5(1):28–38, 2007.

[15] P. Missier, N. W. Paton, and K. Belhajjame. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *EDBT*, pages 299–310. ACM, 2010.

[16] P. Missier, S. Sanket Sahoo, J. Zhao, C. A. Goble, and A. P. Sheth. *anus*: From workflows to semantic provenance and linked open data. In *IPAW*, pages 129–141. Springer, 2010.

[17] L. Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2-3):99–241, 2010.

[18] T. M. Oinn, R. M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, Matthew R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[19] Chun Ouyang, Eric Verbeek, Wil M. P. van der Aalst, Stephan Breutel, Marlon Dumas, and Arthur H. M. ter Hofstede. Formal semantics and analysis of control flow in ws-bpel. *Sci. Comput. Program.*, 67(2-3):162–198, 2007.

[20] Uwe Radetzki, Ulf Leser, S. C. Schulze-Rauschenbach, J. Zimmermann, Jens Lüssem, Thomas Bode, and Armin B. Cremers. Adapters, shims, and glue - service interoperability for *n silico* experiments. *Bioinformatics*, 22(9):1137–1143, 2006.

[21] J. Rothenberg. Ensuring the longevity of digital information. 1995.

[22] Kunal Verma and Amit P. Sheth. Semantically annotating a web service. *IEEE Internet Computing*, 11(2):83–85, 2007.

[23] Patrick Ziegler, Christoph Kiefer, Christoph Sturm, Klaus R. Dittrich, and Abraham Bernstein. Detecting similarities in ontologies with the soqa-simpack toolkit. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *EDBT*, volume 3896 of *Lecture Notes in Computer Science*, pages 59–76. Springer, 2006.

[13] http://www.w3.org/2011/prov/wiki/Main_Page

[14] http://www.myexperiment.org