Computing and Informatics, Vol. 30, 2011, 773–791

WORKFLOW SIMILARITY ANALYSIS

Michał Krzywucki, Stanisław Polak

AGH University of Science and Technology
al. A. Mickiewicza 30
30-059 Krakow
Poland
e-mail: krzywuck@wszib.edu.pl, polak@agh.edu.pl

Communicated by Isabel Campos Plasencia

Abstract. As distributed systems have emerged they become widely adopted by scientific communities and various commercial vendors. Some of them settled architectural models that today are broadly known as grid computing and service oriented architecture. Both of them are base on service and workflow paradigms as a single shareable unit of work. Unfortunately, an increasing number of workflows being brought out has raised a problem of their distribution and management, e.g., search repositories and find workflows similar to the given one in order to increase the efficiency of calculations. In this paper a similar workflow search algorithm based on semantic type comparison has been proposed. In order to evaluate the algorithm usability and precision, an experiment has been conducted that regarded workflow extraction from *Feta* repository. The entire process involved reasoning based on *myGrid* ontologies. The received results were compared to results obtained from other algorithm, based on an analysis of the names of the workflow components using the TD-IDF weight. The described experiment shows that semantics and ontology play significant role in service and workflow representation.

Keywords: Workflow discovery goals, workflow matching algorithms, worklflow similarity based on semantic type comparison, term comparison method versus semantic type comparison method, the TF-IDF method

Mathematics Subject Classification 2000: 68T30, 68T35, 03B70

1 INTRODUCTION

In general, a workflow is considered as a shareable unit of work. It consists of services that are logically connected by control and data flow constraints. Each workflow takes a certain number of input parameters and produces output data or has some defined impact on the environment state.

Implementation of software platforms supporting the concept of workflows and their adaptation to the realities of the target environments is associated with a number of benefits for commercial and scientific organizations, like, e.g.:

1. reduction of the time needed to start work on a new scientific issue – a user wishing to continue experiments in the field does not need to start work from scratch,
2. making the results of research available to the public in the form of workflows allows them to be further used, including possible modifications for new applications.

The use of workflows is associated with the problem of their distribution and management. Each user of distributed systems possessing the ability to build these structures (workflows) needs an infrastructure for their search to find, for example, workflows that are functionally similar to the given one. A similar workflows search system is expected to indicate the corresponding best possible fit based on predetermined criteria. Unfortunately, at present there is no clear similarity quality function which would rank results obtained in terms of their usefulness. In most cases, this process requires direct human intervention. However, it seems that it is possible to identify a few degrees of similarity.

One of them, the easiest to formulate, could be based on the identity relation. For example, two services can be considered highly similar, if it is the same component but installed in various locations and available at different addresses. In the context of workflows they can be logically the same structures differing only in instances of services used. This type of similarity could be used for the purpose of raising the level of workflows availability, or their load balancing.

Probably, another example of the similarity relationship with a lower degree measure might be workflows performing different tasks, but their purpose is the same. In other words, the results of workflows are consistent in terms of semantics. To illustrate this issue it is worthwhile to quote numerical integration algorithms. Having this type of similarity workflows, the increase of the efficiency of calculations and precision of the obtained results could be possible.

The last example of the similarity, being most difficult to assess, is to find the workflow that is sufficiently convergent with a pattern so that it is useful for a reason defined by the user. For instance, a user can search for workflows that use the same service to analyze the scenarios of its use.

The problem of measuring workflows similarity is presented in a few papers. In [3] an algorithm of comparing business workflows that contain complex block

structures such as parallel OR, parallel AND is described. The similarity measure is calculated by analyzing the workflow block structure. The algorithm consists of four steps: conversion of workflow into a block tree, transformation of the tree into a binary tree, construction of vector representation of the binary tree, and the last step is calculation of the distance metric based on the above representation.

In [23] the authors describe workflows similarity methods based on Finite State Automata (FSA). Each workflow is represented as FSA, and then two kinds of similarity approaches: structural automaton similarity, and automaton language based similarity, are considered.

In [4] workflows are represented as business processes dependency graphs. Each graph is converted into a normalized process matrix, and then the metric space distance between the normalized matrices is calculated.

In [20] a method of finding functionally similar workflows is described. The authors assume that naming conventions, used during creating workflow definitions, can be used to measure similarity. Because the results described in this paper are very promising and, besides, the authors do research in the same field as we do, i.e., analysis of the similarity of bioinformatics workflows, we chose the method for further comparisons.

In this paper we present a method of finding similar workflows in terms of semantics.

The rest of the paper is organized as follows: first, workflow discovery goals are presented. In Section 3 the aforementioned method of finding functionally similar workflows is described in more detail. Next we present the method of finding similar workflows based on semantic type comparison. Results of applying this method to compare workflow contained in the *Feta* repository are shown in Section 5. The next section contains comparison of both methods, and the last section concludes the paper.

2 WORKFLOW DISCOVERY GOALS

Despite the fact that the workflow technology is relatively new and new usage scenarios are being developed every day, it is possible to point out some general purposes for workflow discovery and workflow similarity.

The first and the least complex goal is to find a workflow or service that meets specified requirements and invoke it as is [2]. It is not necessary to provide an insight into its construction details unless general purpose of the workflow is satisfactory enough.

The second goal, which tends to be more demanding, is to explore infrastructure resources in order to find workflows or services that are suitable for automated composition matching a given interaction pattern [8]. It is very likely that a process mentioned above will not be performed directly by an end user, but rather by a dedicated software agent. This usage scenario is sometimes called *resource reuse*.

The last goal which undoubtedly is more desired by scientists is called *service repurpose*. There is a stark difference between this approach and those mentioned before. Once a required workflow or service interaction is found, the infrastructure is supposed to provide more detailed information regarding internal control and data flow. Such comprehensive results are not only vital to allow a scientist to analyze and understand the entire process, but it also gives him/her a possibility to make changes in order to achieve new behaviour [21].

3 TERM COMPARISON APPROACH

Bioinformatics is one of the most eager domains to adopt workflow related technologies. In the recent years, many biological databases [5–7] were made accessible via *Web Services* protocols which resulted in migration towards Service-Oriented Architecture (SOA). Most bioinformatics repositories were integrated with graphic design tools like *Taverna Workbench* [21] or *Kepler* [11].

Scientists were given a powerful application that could improve their research by refining service composition. Collaboration was achieved by publishing workflow definition into some kind of a repository which was available for every interested party. Usually it is based on a file server that stores service or workflow descriptors in a tree-like structure. As the number of stored services and workflows useful in a distinct research area is big, creation of new workflows, in many cases, can be replaced by finding existing, similar workflows and then their adaptation to meet the needs.

One of a few algorithms [20] in this field was proposed by researchers from Osaka University. Their work assumed that workflow similarity can be measured by semantics contained by the names of the components like input or output parameters. It assumes that naming conventions, used during creating workflow definitions, can be used to measure similarity. As this approach introduces an admissible level of uncertainty, the TF-IDF *(Term Frequency – Inverse Document Frequency)* filtering method [19] was used to remove irrelevant results. The algorithm was implemented to process *XSCUFL* [17] files shipped as samples from *myGrid* project [16]. Each definition was used as so called "virtual workflow" and matched up with others from the list. Below this algorithm is described in more detail.

Workflow, in the nomenclature adopted in the described algorithm, is represented by a set of literals, and each of them represents input/output parameters, their data types – input/output ports, and tools (local tools and Web Services). In Figure 1 a model of the virtual workflow, which is the input for the algorithm, is shown.

The process of matching the virtual workflow to a given pattern has been divided into three main stages. In the first, literals describing the output parameters are analyzed. The virtual workflow is compared to each item in a workflow repository with a binary function $match(string_1, string_2)$. Provided that $length(string_1) \leq length(string_2)$,

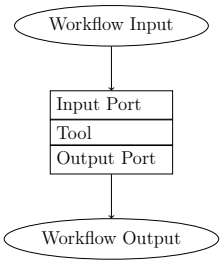


Fig. 1. Virtual workflow model [20]

$$match(string_1, string_2) = \begin{cases} TRUE & \text{if } \frac{length(comsub)}{length(string_1)} \geq 0.5 \wedge \frac{length(comsub)}{length(string_2)} \geq 0.8 \\ FALSE & \text{otherwise} \end{cases}$$

where *comsub* is the longest common substring for $string_1$ and $string_2$, e.g., for the "Medline" string and the "Medline_ID", the *comsub* is "Medline". Both boundary factors were selected by the authors during conducted experiments. If the above function positively recognises at least one pair of literals the entire workflow is qualified for the next stage.

A result set created by the above procedure is narrowed in the second stage through the analysis of literals representing input of the workflow – the above-mentioned function $match(string_1, string_2)$ is being used. The result of the second stage is a set of workflows with similar input and output parameters.

The third phase differs significantly from the previous ones and reaches into the structure of the workflow. The names of input and output literals, and the names of tools related to the services component of compared workflows are considered. An important change is comparing only the most important terms in the virtual workflow and in the potential similar workflow. The weight function is a key element of this phase. Its role is to compute term relevance so uncertain results can be selected and removed from a final set. The idea behind it is solely derived from a statistical measure named TF-IDF. The weight function is expressed as follows: $\omega = tf \times \log(\frac{N}{df})$ where

- tf stands for a number of occurrences of a given term within a single workflow descriptor,
- df is a number of workflows that a specified term occurs in,
- N stands for a total number of workflows in a repository.

Values returned by the function have relative meaning, i.e., the more frequently a given term occurs in a single workflow definition the greater weight it will get computed, whereas df value has substantially the opposite meaning. The more

often a specified term is used in other workflow descriptors the lesser weight will be applied.

The conducted experiments raised the question regarding algorithm precision. As there is no formal definition of functional workflow similarity, the result had to be validated manually. The authors have assumed that similarity criteria are met when extracted workflows can be used interchangeably with a slight modification of an internal structure. This approach has let authors achieve 99.2 % result precision.

Even though the extraction method seems to be very promising, several obstacles are worth mentioning here. All the examined workflow descriptors were taken from myGrid samples which could let us expect more issues, as the number of workflows increased. Secondly, the terms used to measure similarity were derived from a single usage domain. It is very likely that once workflow technology is spread to other areas, more naming collisions and term mismatches will occur.

4 SEMANTIC APPROACH

Ontology [9] is a formal representation of the knowledge in Semantic Web applications [1], in semantic search engines like, e.g., the *Feta* system described below, and in other intelligent applications [22].

4.1 The *Feta* Repository

Taverna Workbench widely adopted workflow composition and enactment tool, especially with regard to scientific communities. It can be used to visually compose interaction patterns and save them as a XSCUFL descriptor to let other researchers make use of it. One of the most interesting features is that the workbench can be enriched with the *Feta* [13] system (the *Feta* plugin). It allows users to browse service repository and import desired workflow into their own workspace. A draft view of the architecture of the *Feta* system is presented in Figure 2 – it shows information flow among its individual software components, as well as between the components and users.

Unfortunately, the overall feature set is pretty limited[1]; however, internal repository data model can be valuable for the sake of the workflow similarity analysis.

The repository structure adopts recent *Semantic Web* achievements. It stores *RDF* [14] triples which follow constraints specified by the *myGrid* ontology.

The myGrid ontology [15] is built from two separate sub-ontologies that serve different purposes. The first defines basic predicates which are used to point out service building elements. The second, named domain ontology, provides wide hierarchy for data and algorithm types. Those are associated with parameters and tasks in order to attach metadata. A sample service description is described in Figure 3.

[1] At the time of writing this paper it was impossible to form and send custom queries to the repository. A web service that allowed to serialize entire content into RDF file was used to prepare data for further experiments.

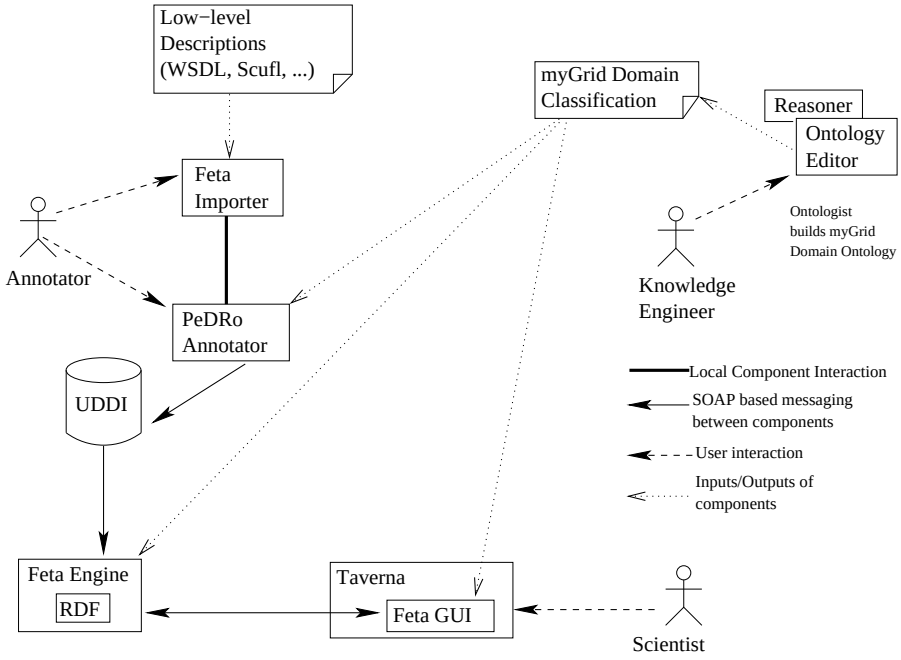


Fig. 2. The *Feta* system architecture [2]

Feta semantic service model consists of blank nodes that lead to semantic types. According to the described data model, *Feta* service can contain one or more operations which are associated with input and output parameters.

Parameters and tasks can be given textual names and descriptions. These literals are often used to store names which are imported from other formats like XSCUFL files.

4.2 Workflow Extraction Algorithm

Feta service model contains much more valuable metadata than plain XSCUFL descriptors. It can be used as a more reliable and more comprehensive similarity marker.

The idea of the first two phases of the proposed algorithm is fairly similar to its TF-IDF based predecessor. Invalid workflow pairs are eliminated by output parameter matching. Subsequently, input parameters are compared. However, there are two significant differences. Comparison function makes use of associated semantic type identifiers instead of literals. Secondly, there is no need to employ fuzzy term matching as ontology guarantees unambiguity.

There is a compelling dissemblance with regard to the third phase. Since *Feta* semantic model hides internal workflow structure by covering it with a semantic

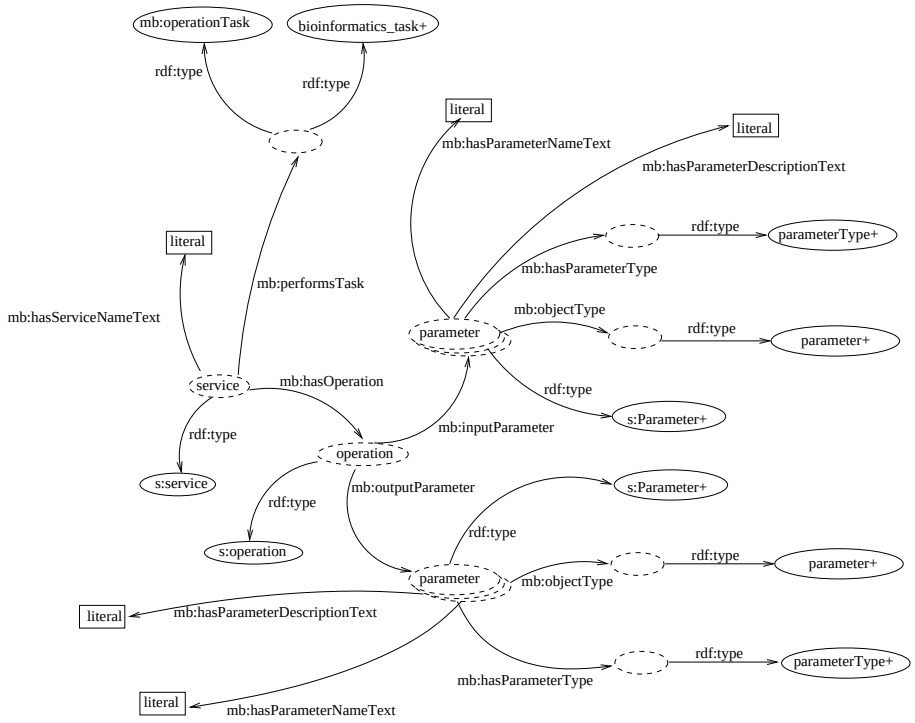


Fig. 3. myGrid semantic service model [12]

task concept, it does not make sense to measure TF-IDF weight. Fixed comparison of task semantic types is used instead.

Since *myGrid* ontologies provide deep type hierarchy, it was possible to propose a significant refinement to the matching algorithm. Given that the *subclassOf* relation is commonly used to link general and more specific semantic types, it is likely that workflows annotated with semantic types within the same type hierarchy will maintain the same relation.

For the sake of example, let us assume that a given workflow returns an output parameter annotated with *nucleotideSeqence* type. It is likely that another workflow that returns *DNAsequence* can be considered similar as its output fulfills the same requirements[2] specified by *myGrid* ontology.

This approach involving reasoning by types was applied to three similarity criteria: input, output and a task being performed. If all requirements were satisfied, two workflows were considered similar.

[2] General semantic types can have limitations against, i.e., class predicates, relations, etc. More specific semantic types have to fulfill the same limitations (requirements) too.

5 EXTRACTION RESULTS

In order to evaluate the proposed algorithm, an experimental application *Fetawf* was implemented. Its basic task is to find similarity among stored services. The entire content of *Feta* repository was serialized into an RDF file and used as input to build semantic model based on Jena Semantic Web Framework [10]. Service descriptors and myGrid ontology files were merged together in order to prepare data model for further experiments. The data flow between components of the *Feta* repository and the *Fetawf* application is shown in Figure 4. At the time of conducting this research the *Feta* repository was characterised by the following number of elements – see Table 1.

Number of the *RDF* triplets	44 519
Number of the *Feta* descriptors	601
References to the XSCUFL descriptors	388
Semantically described	506
Number of I/O parameters	2 448
Number of references to the WSDL descriptors	585
File size of the RDF set	7MB

Table 1. Quantitative characteristics of the elements of the *Feta* repository

Every workflow definition structure was compared to each other in order to check whether similarity criteria are met. Technically speaking, comparison process was accomplished by invoking SPARQL [18] queries combined with direct Jena API calls onto underlying repository. If a given pair passed the similarity test, it was considered as a successful extraction and saved in a result list.

Figure 5 shows workflow comparison results after the first stage regarding output comparison. Repository elements were numbered and put into a matrix (a similarity matrix). Each row or column refers to a different workflow. The squares that are placed at the intersections represent a successfully identified similarity between two given workflows.

The diagonal line built from the squares refers to similarity of a workflow to itself. It is worth mentioning that similarity squares are placed symmetrically around the diagonal, which means that the comparison method is a symmetric relation in this case.

Nevertheless, there are several exceptions marked as circles. Those pairs were considered similar with a help of type reasoning. In this case only one way similarity relation is satisfied.

To have a better understanding of the issue let us have a look at the following example described in Figure 6. Given that the task associated with workflow 2 is an indirect descendant of a task associated with workflow 1 it is assumed that workflow 1 is similar to workflow 2 with regard to the task type. Having said that though, there are no prerequisites to believe that an opposite relation is satisfied.

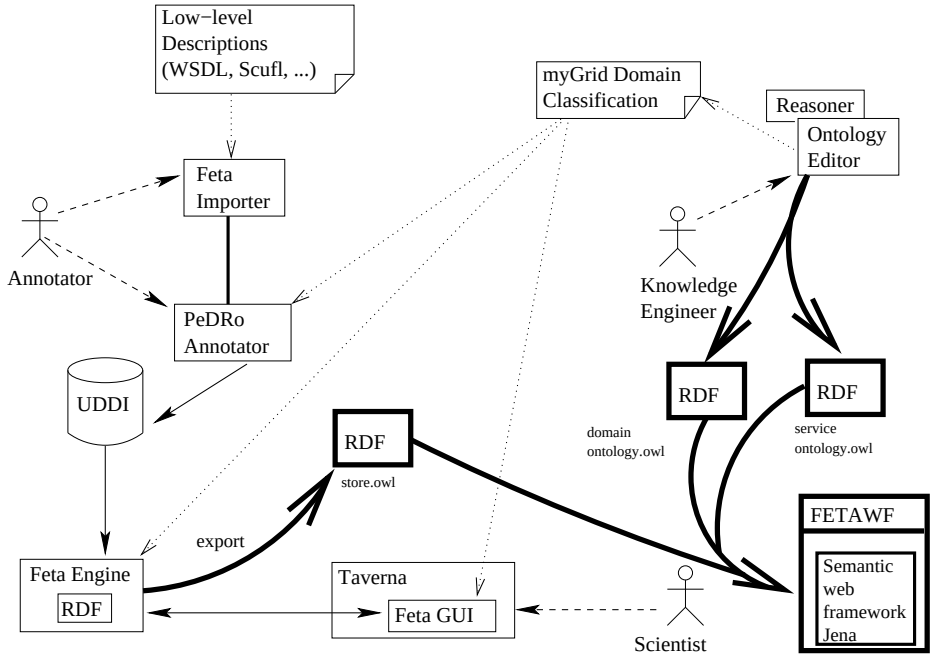


Fig. 4. Data flow between components of the *Feta* repository and the *Fetawf* application

Even though it may seem probable it is not possible to state that workflow 2 is similar to workflow 1 with regard to algorithm type.

The number of pairs of similar workflows found after the first stage (phase) is presented in Table 2. As one can see, using inference during output parameter matching phase allowed to extract additional 336 matchings. After the last stage we received a result set consisting of 132 pairs[3] of similar workflows.

Number of matchings	1 376
Number of inferred	336

Table 2. The number of workflows found by the algorithm after the output parameter matching stage

6 METHOD COMPARISON AND QUALITY ESTIMATION

At the moment of writing this paper it seemed that it was impossible to verify the algorithm correctness automatically. Results had to be checked manually. Hence a bioinformatics scientist was asked to evaluate whether a workflow pair could be

[3] Workflows similar to themselves are not taken into account

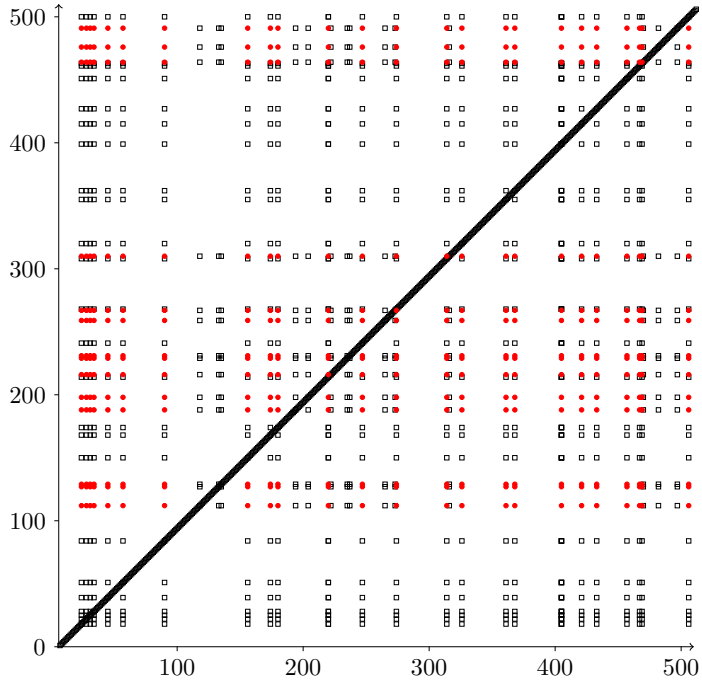


Fig. 5. Output matching asymmetry (the axes contain the corresponding numbers of workflows)

considered similar. A possible answer set was purposely limited to three options (subsets):

- workflows that are definitely similar and can be used interchangeably with a slight modification,
- workflows whose similarity is questionable, which means that interchangeable usage requires more complex recomposition,
- workflows that are certainly different and should not be considered similar.

The cardinality of the subsets is presented in Table 3. Method correctness evaluation results are presented in Figure 7. It shows that only 20 % of all extracted workflow pairs are definitely wrong. It means that if we apply the same quality criteria as with regard to TF-IDF algorithm, the overall method correctness is equal to 80 %. It is obvious that results may differ if the entire process is provisioned with data from other sources. Nevertheless, the *Feta* repository was probably the only single database that contained semantically annotated descriptors at the time of writing this paper.

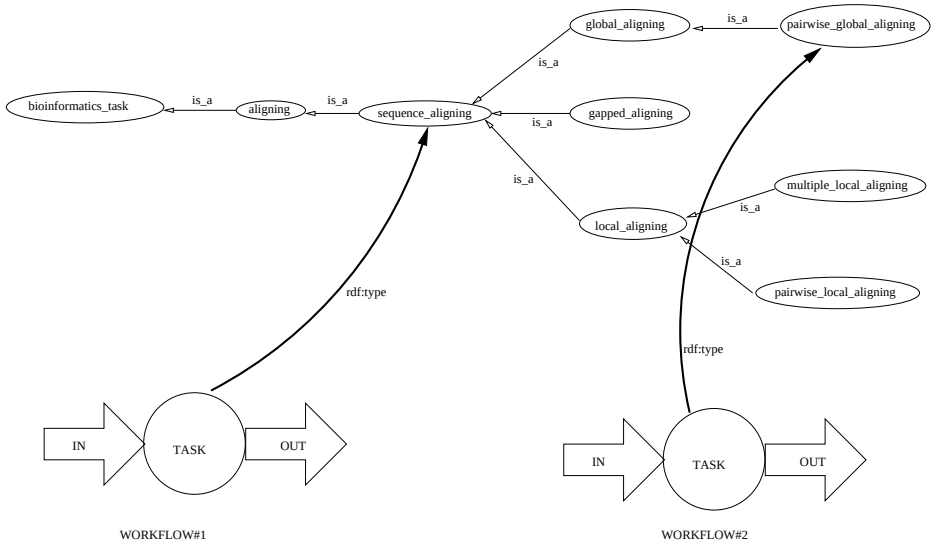


Fig. 6. Similarity relation asymmetry

Subset	Cardinality
definitely similar	75
definitely dissimilar	26
questionable similarity	31

Table 3. Analysis of results of semantic method extraction

In order to correlate extraction results given by two approaches, an attempt to compare the semantic method with the TF-IDF workflow algorithm described earlier was performed.

Because workflow descriptors stored in the *Feta* repository, as compared with the XSCUFL files mentioned in Section 3, were less detailed, the TF-IDF was slightly changed. The terms that were used for lexical comparison were taken from literals associated with given parameters and task nodes of a service model. The third stage related to TF-IDF filtering algorithm was simplified as there is only one task name used for each service descriptor.

Workflow extraction results are shown in Figure 8. Term based approach extracts much more similar pairs of workflow than the semantic based one. What is most striking is the fact that only 41 pairs were extracted by both methods. The above results indicate that it is easier to meet the criteria of similarity of names than the assigned types. The studies have shown that this is mainly caused by the following factors:

- not all services had assigned semantic types – some of them were added to the repository due to automatic import from other sources,

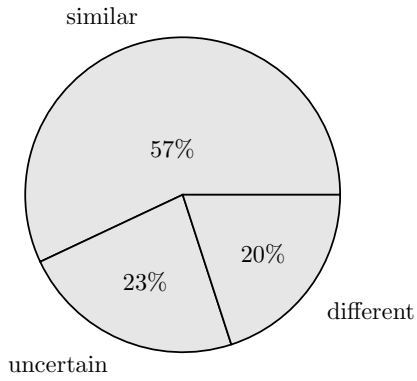


Fig. 7. Correctness percentage distribution

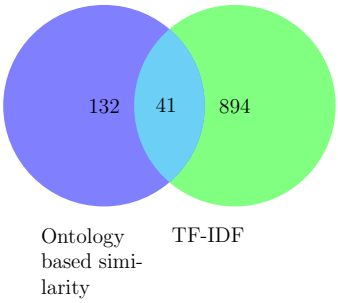


Fig. 8. Methods results comparison

- use of the same, common literals to name the parameters or tasks, (e.g. *report*, *arg*, *data*, *info*)
- using very similar names in terms of similarity function (e.g. *DNA*, *RNA*).

Another very important difference can be seen from the similarity matrix. Algorithm based on the semantic types forms result in linear shapes (see Figure 9). This means that groups of services, which are to each other similar, were found. This is a result of classification carried out in the annotation process of services.

In the case of the term based algorithm such regularity is not seen (see Figure 10). Showing the pair of matching points services are freely scattered over the whole surface of the chart. This allows to conclude that these matches are more random. One can venture to say that the term based algorithm returns workflows similar in terms of used naming conventions and composition habits. Assuming that all the authors of workflows are motivated by the same guidelines during creating a definition, one can count on satisfactory results of finding similar workflows with this method.

Determining the similarity in the semantic based method is implemented at a slightly different plane. Classification of individual components of definition using a consistent glossary allows determination of corresponding workflows in terms of their destination.

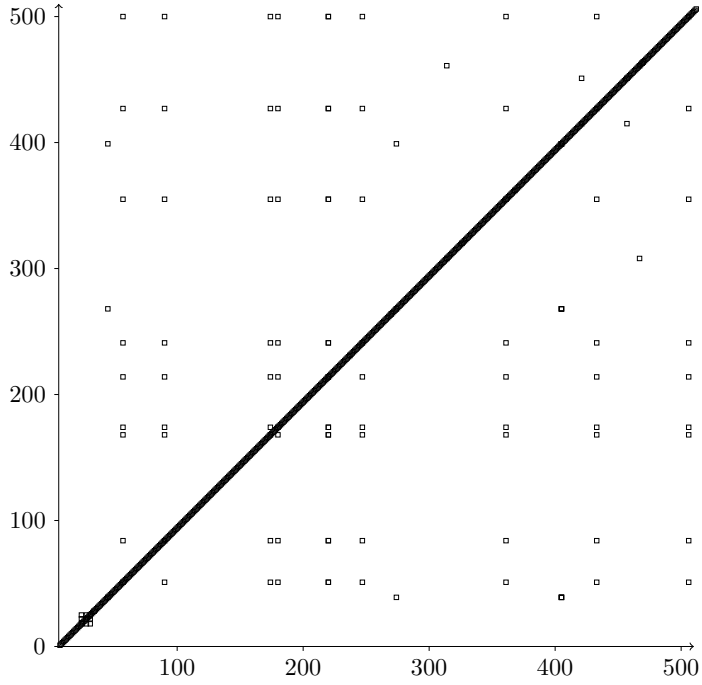


Fig. 9. Result similarity matrix for the semantic based approach

The term based approach led to naming collision and mismatches which unfortunately could not be avoided by filtering in this case.

The semantic comparison relies mostly on annotations attached during service classification. This is a process that requires overall cognition of entire repository resources and remains tightly related to knowledge engineering, whereas the term based approach makes use of naming conventions that are very likely to differ for each research group.

To better understand the difference between these two algorithms, it is worth analysing example pairs found by both methods and pairs identified only by one of them.

Table 4 contains characteristics of two example workflows: *transeq* and *backtranseq*, i.e., the names and the types of the input / output parameters and tasks, as well as short workflow description. This matching was identified by both methods as a result of both semantic type matching and big similarity of parameter names.

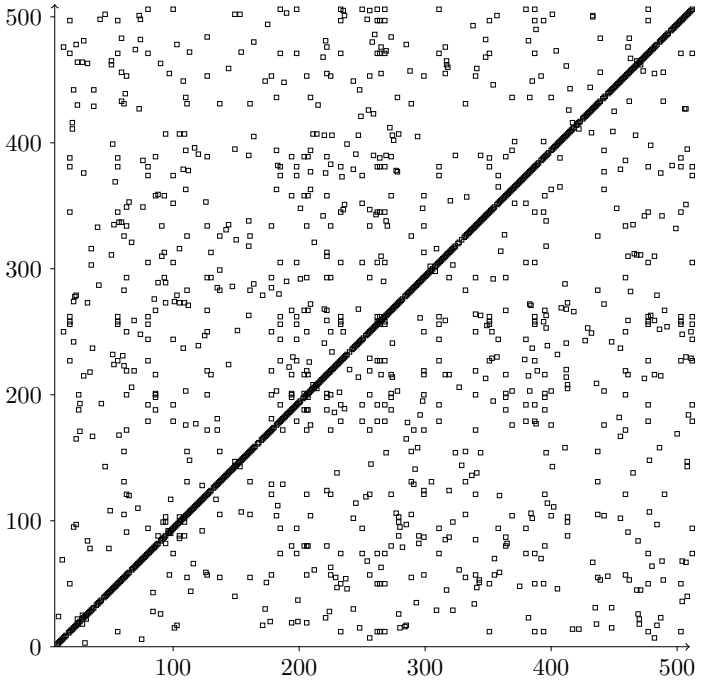


Fig. 10. Result similarity matrix for the term based approach

One key difference are the names of the tasks: *transeq* and *backtranseq*, but these names are in the TD-IDF method tolerance limits.

transeq		***backtranseq***	
INPUT			
NAME	TYPE	NAME	TYPE
sequence_usa	mb:single_sequence_format	sequence_usa	mb:single_sequence_format
sequence_direct_data	mb:biological_sequence	sequence_direct_data	mb:biological_sequence
sbegin	–	sbegin	–
sformat	–	sformat	–
...	...	...	...
OUTPUT			
outseq	mb:biological_sequence	outseq	mb:biological_sequence
report	–	report	–
detailed_status	–	detailed_statsu	–
TASK			
transeq	mb:translating	backtranseq	mb:translating
DESCRIPTION			
Returns modified sequence		Returns sequence with masked features	

Table 4. Example pair of similar workflows found by both methods

Table 5 contains matching done only by the semantic method. In spite of a big lexical similarity of input and output parameters, this pair was rejected because of the big difference of task names, i.e., *cutseq* and *maskfeat*.

cutseq		***maskfeat***	
INPUT			
NAME	TYPE	NAME	TYPE
sequence_usa	–	sequence_usa	–
sequence_direct_data	mb:biological_sequence	sequence_direct_data	mb:biological_sequence
sbegin	–	sbegin	–
sformat	–	sformat	–
...	...	...	...
OUTPUT			
outseq	mb:biological_sequence	outseq	mb:biological_sequence
report	–	report	–
detailed_status	–	detailed_status	–
TASK			
cutseq	mb:removing	maskfeat	mb:removing
DESCRIPTION			
Returns modified sequence		Returns sequence with masked features	

Table 5. Example pair of similar workflows identified only by the semantic method

The last collation (see Table 6) shows one of the pairs rejected by the semantic algorithm, but accepted by the algorithm based on the TD-IDF method. As one can see, ontological types describing output parameters and tasks (of these two workflows) are different. In spite of semantic types dissimilarity, lexical similarity has been taken into account, mainly because of naming conventions. According to them, the *searchSimpleAsync* task name and the *analyzeSimpleAsync* task name are similar.

searchSimpleAsync		***analyzeSimpleAsync***	
INPUT			
NAME	TYPE	NAME	TYPE
query	mb:biological_sequence	query	mb:biological_sequence
database	–	–	
program	–	–	
OUTPUT			
Result	mb:BLAST_report	Result	mb:multiple_sequence_alignment_report
TASK			
searchSimpleAsync	mb:sequence_aligning	analyzeSimpleAsync	mb:multiple_local_aligning
DESCRIPTION			
Execute BLAST asynchronously with program, database and a query sequence.		Execute ClustalW asynchronously with multiple sequences.	

Table 6. Example pair of similar workflows identified only by the TF-IDF method

7 CONCLUSIONS

The described experiment shows that semantics and ontology play significant role in service and workflow representation. The provided metadata may be valuable for workflow comparison and extraction. It brings several benefits like disambiguation and eliminates undesired naming collisions.

Semantic approach allows to scale the problem domain. It is very likely that workflows which were designed for different purposes can be easily merged to form more comprehensive repositories. In accordance with semantic web and ontologies

assumptions each workflow is uniquely identified so there is no risk that descriptor elements may collide.

The term based approach fails at this point. It extracts too many workflows that follow similar naming convention but refer to different algorithms. Even though statistical measures are used to eliminate uncertain data it cannot comprehensively avoid the problem.

Reasoning is another aspect that makes us believe that most usable features are yet to be implemented. According to ontologies standards, metadata can be easily extended and combined with more sophisticated reasoning rules than type hierarchy.

Acknowledgements

We would like to thank Dr. Tomasz Arodź and Prof. Jacek Kitowski from Institute of Computer Science AGH University of Science and Technology in Kraków for their help. In particular, Dr. Tomasz Arodź is acknowledged for verification of the obtained results. The work was funded by AGH under grant 11.11.120.865.

REFERENCES

[1] Andrejko, A.—Bieliková, M.: Comparing Instances of Ontological Concepts for Personalized Recommendation in Large Information Spaces. Computing and Informatics, Vol. 28, 2009, No. 4, pp. 429–452.

[2] Alper, P.: User-Oriented Semantic Service Discovery. Ph. D. thesis, University of Manchester, 2004.

[3] Bae, J.—Caverlee, J.—Liu, L.—Yan, H.: Process Mining by Measuring Process Block Similarity. In: Business Process Management Workshops, number 4103 in Lecture Notes in Computer Science, pp. 141–152, Springer Berlin/Heidelberg, 2006.

[4] Bae, J.—Liu, L.—Caverlee, J.—Zhang, L. J.—Bae, H.: Development of Distance Measures for Process Mining, Discovery and Integration. International Journal of Web Service Research, Vol. 4, 2007, No. 4, pp. 1–17, 2007.

[5] The BioMoby project web site. Available on `http://www.biomoby.org/`.

[6] EBI: Web Services for EMBOSS-6.1.0 applications. Available on `http://www.ebi.ac.uk/soaplab`.

[7] The Emboss software package web site. Available on `http://emboss.sourceforge.net/`.

[8] Goderis, A.: Workflow re-use and discovery in bioinformatics. Ph. D. thesis, University of Manchester, 2008.

[9] Gruber, T.: The definition of ontology. Available on `http://tomgruber.org/writing/ontology-definition-2007.htm`.

[10] The Jena Semantic Web Framework web site. Available on `http://jena.sourceforge.net`.

[11] The Kepler project web site. Available on `https://kepler-project.org/`.

[12] Krzywucki, M.: Workflow Similarity Research. In Polish. M. Sc. thesis, AGH-University of Science and Technology, 2009.

[13] Lord, P.—Alper, P.—Wroe, C.—Goble, C.: Feta: A Light-Weight Architecture for User Oriented Semantic Service Discovery. University of Manchester, UK, May 2007.

[14] McBride, B.: RDF Primer W3C Recommendation. 10 February 2004. Available on http://www.w3.org/TR/rdf-primer/.

[15] The myGrid ontology web site. Available at http://www.mygrid.org.uk/tools/service-management/mygrid-ontology/.

[16] The myGrid project Home Page. Available on http://www.mygrid.org.uk/.

[17] Oinn, T.: Xscufl Language Reference. 7th April 2004. Available on http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html.

[18] Prudhommeaux, E.—Seaborne, A.: Sparql Query Language for RDF W3C Recommendation. 15 January 2008. January 2008. W3C Consortium.

[19] Salton, G.—Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management, Vol. 24, 1988, No. 5, pp. 513–523.

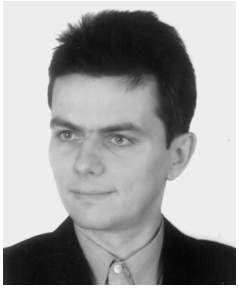
[20] Seo, J.—Seno, S.—Takenaka, Y.—Matsuda, H.: Retrieving Functionally Similar Bioinformatics Workflows Using TF-IDF Filtering. April 2007. School of Information Science and Technology, Osaka University.

[21] The Taverna workbench project website. Available on http://taverna.sourceforge.net/.

[22] Ying, W.—Sujanani, A.—Ray, P.—Paramesh, N.—Lee, D.—Bhar, R.: Design and Development of Financial Applications Using Ontology-Based Multi-Agent Systems. Computing and Informatics, Vol. 28, 2009, No. 5, pp. 635–654.

[23] Wombacher, A.—Rozie, M.: Evolution of Workflow Similarity Measures in Service Discovery. In: M. Schoop, C. Huemer, M. Rebstock, and M. Bichler (Eds.), Konferenz im Rahmen der Multikonferenz Wirtschaftsinformatik, 2006, Vol. P-80, pp. 57–71, Bonn, Germany, February 2006. Gesellschaft fuer Informatik.

Michał Krzywucki received the B. Sc. degree in Computer Science from the School of Banking and Management in Kraków in 2006. In 2009 he received the M. Sc. degree from the AGH – University of Science and Technology in Kraków. He is currently a Project Manager at the School of Banking and Management. His research interests include software architectures, lean software development methodologies, artificial intelligence, semantic web, knowledge processing, and cloud computing.



Stanisław Polak is an Assistant Professor of the AGH – University of Science and Technology (UST) in Kraków. He received the M. Sc. and Ph. D. degrees from AGH – UST in 1993 and 2003, respectively. His research interests include parallel programming and computing, semantic description, the XML language, and Web technologies.