Journal of Computational Interdisciplinary Sciences (2011) 2(1): 23-35

Printed version ISSN 1983-8409 / Online version ISSN 2177-8833
http://epacis.org

Similarity-based workflow clustering

Vítor Silva[1], Fernando Chirigati[1], Kely Maia[1], Eduardo Ogasawara[1,3], Daniel de Oliveira[1], Vanessa Braganholo[2], Leonardo Murta[2] and Marta Mattoso[1]

Manuscript received on August 12, 2010 / accepted on January 28, 2011

ABSTRACT

Scientists have been using scientific workflow management systems (SWfMS) to support scientific experiments. However, SWfMS expect a modeled workflow to be represented on its workflow language to be executed. The scientist does not have an assistance or guidance to obtain a modeled workflow. Experiment lines, which are a novel approach to deal with these limitations, allow for the abstract representation and systematic composition of experiments. Since there are many scientific workflows already modeled and successfully executed, they can be used to leverage the construction of new abstract representations. These previous experiments can be helpful by identifying scientific workflow clusters that are generated according to similarity criteria. This paper proposes SimiFlow, which is an architecture for similarity-based comparison and clustering to build experiment lines following a bottom-up approach.

Keywords: scientific workflow, clustering, similarity.

1 INTRODUCTION

Most of the existing scientific experiments need to process and transform large volume of data, which may be an unviable task to be accomplished without computational support. In many possible scenarios, scientists execute simulations to validate their hypothesis. These simulations depend on the use of computational mechanisms that may be very complex. Scientific workflows are an abstraction that allows the specification of those experiments in a structured manner using a flow of programs, services and data to produce a final result (Deelman et al. 2009).

As the composition of programs to model a coherent workflow become complex (many programs invocations, additional data transformations steps, need for loops and decision points), it is necessary to provide ways to support scientist in composing these workflows (McPhillips et al. 2009). Cavalcanti et al. (2005) propose the use of different abstraction levels to compose scientific workflows as a way to cope with this complexity. Cavalcanti et al. (2005) grade the abstraction levels in two different categories: abstract and concrete. Following their definition, an abstract workflow is the representation of a chain of activities without specifying its mapping to execution resources, while concrete workflows connect the execution of activities with some computational resources. Concrete workflows are usually specified (and executed) in Scientific Workflow Management Systems (SWfMS). However, scientists usually focus their efforts in the conceptual level of the experiment. Due to that a new abstraction level is needed: the conceptual one. A conceptual workflow is the chain of activities that specifies what needs to be done without saying how it must be done (Ogasawara et al. 2009b). The conceptual level is exclusively related to concepts that describe each activity of the workflow. However, existing SWfMS are closely related to con-

Correspondence to: Eduardo Ogasawara – E-mail: eogasawara@cefet-rj.br
[1]Federal University of Rio de Janeiro – UFRJ. – E-mails: {silva,fernando_seabra,ogasawara,danielc,marta}@cos.ufrj.br / kely.maia@dcc.ufrj.br
[2]Fluminense Federal University – UFF. E-mails: vanessa@ic.uff.br / leomurta@ic.uff.br
[3]Federal Center of Technological Education – CEFET-RJ.

crete workflows and do not provide ways to specify conceptual workflows and this type of abstract specification is an open and important issue (Mattoso et al. 2010).

In order to bridge the gap of composing scientific workflows using levels of abstraction, the concept of experiment lines (Ogasawara et al. 2009b) was proposed. The concept of experiment lines aims at minimizing the complexity of scientific experiments composition by defining families of experiments that share common activities. Experiment lines represent a scientific workflow in a conceptual level and can be considered as a meta-experiment, thus representing important information such as optional activities and variability.

There are two different approaches to model an experiment line: top-down and bottom-up. In a top-down approach, from the specification, the user is able to model conceptual and abstract workflows and, then, generate concrete workflows based on the previously modeled conceptual and abstract workflows. This process is called derivation. The top-down approach may be used when scientists are specifying new experiments and previously modeled concrete workflows are not available to be reused. On the other hand, experiment lines also allow workflow composition using a bottom-up approach, i.e., the conceptual and abstract workflows can be modeled from previously imported concrete workflows.

In bottom-up approach scientists have many concrete workflows already modeled and do not have the identification of the conceptual workflow for each one of the concrete workflows. Indeed, a large amount of concrete workflows may possibly be derived from a single conceptual workflow and scientists may not be aware of this fact. In this way, if we could compare pre-existing concrete workflows, we would be able to structure the knowledge of the experiment, thus allowing for executing a bottom-up approach to model experiment lines. However, there is a need of a set of primitives to calculate the similarity between these workflows and to cluster these workflows based on the similarity to promote a bottom-up approach.

This paper presents SimiFlow (Silva et al. 2010), an architecture that focuses on comparing and clustering workflows based on similarity. SimiFlow aims at supporting the modeling of experiment lines following a bottom-up approach. The main goal of this work resides in the process of workflow comparison and clustering without concerning about the modeling of experiment lines. This architecture was implemented in the GExpLine tool (Oliveira et al. 2010) that is responsible for managing experiment lines.

This paper is organized in four sections besides this introduction. Section 2 introduces the SimiFlow architecture. Section 3 discusses experimental results. Related work is presented in Section 4 and finally Section 5 concludes the paper and points to future work.

2 SIMIFLOW

This section presents SimiFlow and its components. SimiFlow was designed and developed aiming at comparing and clustering scientific workflows based on their similarity. This section is divided into four subsections. Subsection 2.1 presents the conceptual architecture. Subsections 2.2, 2.3, and 2.4 explain the components and algorithms developed to compare and cluster a set of concrete scientific workflows.

2.1 Architecture

The architecture of SimiFlow was designed aiming at analyzing and grouping scientific workflows from a set of concrete workflows based on the similarity of each pair within this set. Once the similarity is calculated, SimiFlow is capable of generating clusters of concrete workflows. The elements of each one of these clusters share common characteristics and elements, thus representing a family of workflows. As the concept of experiment lines aims at representing families of workflows we can benefit from the SimiFlow architecture to model experiment lines based on a specific set of concrete workflows.

Since there are many types of algorithms to calculate similarity, the SimiFlow architecture is extensible to provide ways to change algorithms and methods to compare and cluster workflows. In this way, SimiFlow architecture was conceived by using the design patter *strategy* (Gamma et al. 1994) which is focused on the dynamic change of algorithms developed for a specific application. Based on this *strategy* pattern it is possible to define a group or family of algorithms and then encapsulate them into objects.

Three different cartridges (Birsan 2005) were developed to be used by SimiFlow: a cartridge to import workflows, a cartridge to compare workflows, and a cartridge to cluster workflows based on the calculated similarity. The first cartridge is responsible for obtaining concrete workflows from external sources and then storing important structural information of these workflows in the GExpLine database schema. The second cartridge compares pairs of scientific workflows and calculates the similarity between them. The last cartridge clusters the workflows based on the calculated similarity. Figure 1 presents the proposed architecture and the different cartridges. The following subsections explain each one of the cartridges.

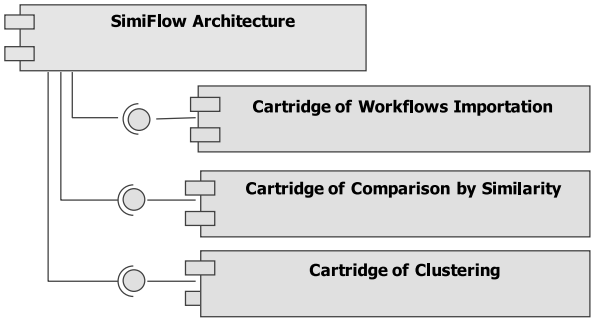

Figure 1 – SimiFlow architecture.

2.2 Workflow import

The cartridge to import scientific workflows provides ways to store important structural information in the GExpLine schema related to scientific workflows specified in the Taverna (Oinn et al. 2004), VisTrails (Callahan et al. 2006), and Kepler (Altintas et al. 2004) SWfMS. These SWfMS allow scientists to specify concrete workflows using specific formats based on XML. Those XML files are then imported and analyzed by SimiFlow, which is capable of mapping the information represented in the XML file to the relational database schema of GExpLine. Figure 2 presents a fragment of a XML file generated by Taverna and the mapping of XML elements to the classes of the GExpLine schema (Oliveira et al. 2010).

In order to evaluate different algorithms for comparing and clustering workflows based on similarity, it is important to work with an expressive number of workflows. To achieve this goal, the set of workflows imported to the GExpLine schema was gathered from the website myExperiment (Goble & Roure 2007). The myExperiment site provides a repository of scientific workflows. At December 2009, the myExperiment had 756 Taverna workflows including 463 public workflows.

2.3 Comparison

The goal of the comparison cartridge is to analyze each pair of scientific workflows in the set of concrete workflows and calculate a similarity value for each pair. Although the architecture is ready to support different cartridges for comparison, this paper presents only one cartridge, which was used as an initial implementation.

The basic cartridge that compares scientific workflows and calculates the similarity between them executes a comparison algorithm that consists of two phases. The first phase (penalty phase) structurally compares each pair of scientific workflows. The algorithm initially assumes that each pair of workflow elements from the comparing workflows is equal. The algorithm then starts observing each pair of workflow elements using different criteria. Each difference observed is considered a penalty and it is used to compute the similarity. The second phase (similarity phase) analyzes all penalties generated for each pair of workflow elements and summarizes the final similarity for each pair of workflows.

2.3.1 Penalty phase

The first phase of the comparison algorithm is the penalty phase. As discussed before, this phase is responsible for comparing all workflow elements of each pair of scientific workflows of the initial set of workflows. Let us suppose that we are comparing activities of workflows W_1 and W_2. Each activity of W_1 is compared with each activity of W_2 considering a specific set of characteristics of each activity (that we call elements and are further detailed following). The algorithm performs many comparisons of tuples like (A_i, A_j) where A_i is an activity of W_1 and A_j is an activity of W_2. This comparison activity considers the following elements to compare: name of the activity, type of activity, port, relationship, and internal structure (subworkflow or XML definition of the activity). The comparison process of each element generates a penalty value: PnA (associated to the name of the activity), PnT (associated to the type of the activity), PnP (associated to ports), PnR (associated to relationship), and $PnIS$ (associated to the internal structure of the activity) that is composed by $PnSw$ (associated to subworkflow) or PnX (associated to the XML definition of the activity). For each comparison, a penalty is applied and the final penalty for each pair of activities (A_i, A_j) is defined as:

$$\begin{aligned} PnTotal_{A_i,A_j} = PnA_{A_i,A_j} + PnT_{A_i,A_j} \\ +PnIS_{A_i,A_j} + PnP_{A_i,A_j} + PnR_{A_i,A_j} \end{aligned} \quad (1)$$

and:

$$PnIS_{A_i,A_j} \begin{cases} PnSw_{A_i,A_j}, \\ \text{if } A_i \text{ or } A_j \text{ is a subworkflow} \\ PnX_{A_i,A_j}, \\ \text{if } A_i \text{ and } A_j \text{ are not subworkflows} \end{cases} \quad (2)$$

The penalty values vary according to the type of element being considered. The maximum penalty values are shown in Table 1. Note that the maximum penalties used in this paper are part of an initial study and these values may be fine-tuned in

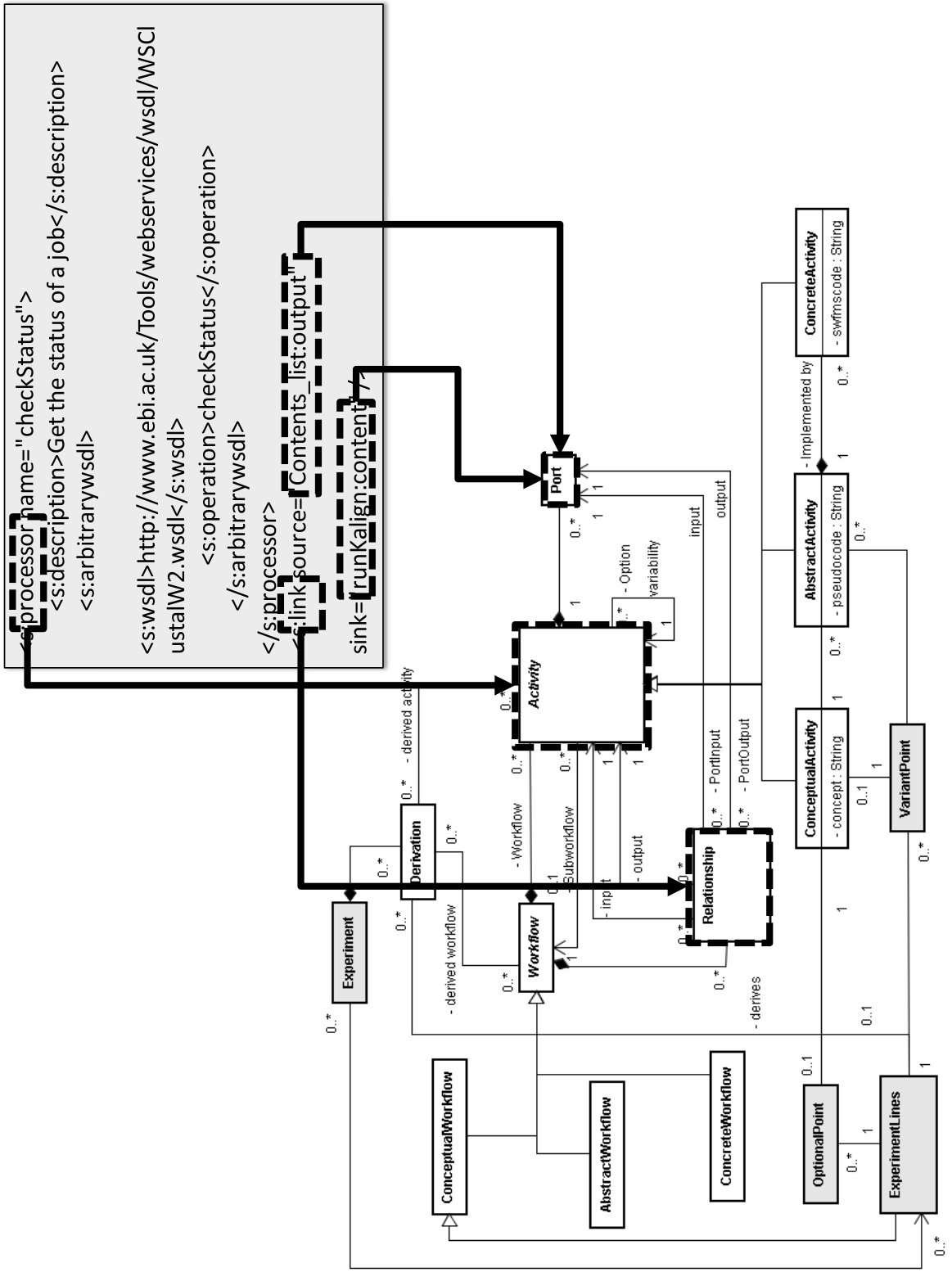

Figure 2 – Mapping workflow structure to GExpLine schema.

the future to improve SimiFlow. However, these values are the maximum penalties applied for each criteria and the computed penalty depends on the number of differences found. Throughout this section, each criterion is described in detail.

Table 1 – Element maximum penalties.

Element	Maximum penalty
Name of activity	512
Type of activity	1024
Port	128
Relationship	64
Internal structure (XML definition or subworkflow)	256

Name of activity

This is the simplest comparison element. The algorithm analyzes the names of the activities A and A' and if they are different it applies the maximum penalty $PnA_{A,A'} = 512$. Figure 3 presents a trivial example of name comparison between activity A and A'. In this example the penalty applied was 512.

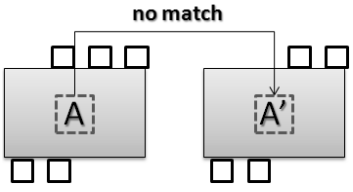

Figure 3 – Activity name comparison.

Type of activity

The second comparison element is the type of activity. Each activity in a scientific workflow (independent of SWfMS) has a specific type. For example, there are Web Services activities, command lines activities, grid services activities and so on. The algorithm analyzes the types of both activities A and A' and if they are different it applies the maximum penalty $PnT_{A,A'} = 1024$. Figure 4 presents an example of type comparison between activities A an A'. These activities have different types and, in this case, the penalty applied was 1024.

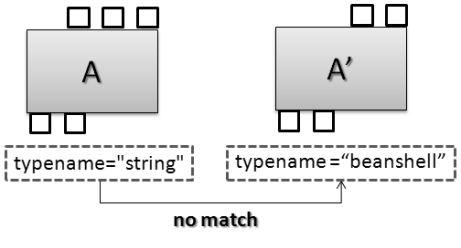

Figure 4 – Activity type comparison.

Port

The third comparison element is the set of ports of the activity. Different from the two previous elements, the penalty applied to port comparison is proportional to the number of different ports between a pair of activities. It considers three types of ports: input ports (IP), output ports (OP), and multiports (MP) that can act either as input and output ports and whose behavior is not known *a priori*. Let us explain with a theoretical example. Figure 5 presents a comparison between activities A and A'. A has 3 input ports named X, Y, and W while A' has 2 input ports named X and Z. A has also 2 output ports and A' has 2 output ports as well.

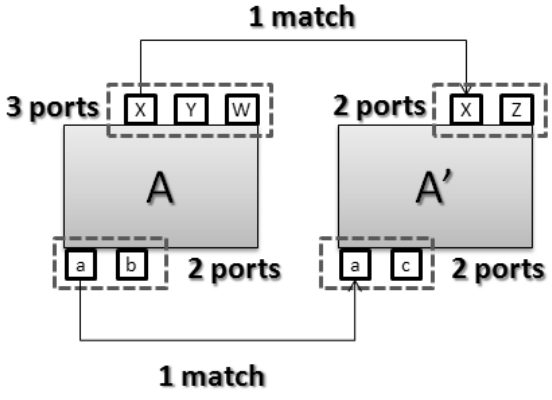

Figure 5 – Activity port comparison.

The first part of the algorithm compares each input port of A with all input ports of A'. If the name and the type of each pair of ports are equal then it is computed as one match. This process is repeated until all ports are compared. In the case of the example of Figure 5 there is only one match (port X) and the other three ports do not have a correspondent port. In this way, the penalty applied here must be proportional to the number of different ports. The algorithm calculates a proportional factor α_i for the input ports. The proportional factor α_i of this example is calculated as follows:

$$\alpha_i = \frac{(IP\ of\ A + IP\ of\ A') - (2 \times matches)}{(IP\ of\ A + IP\ of\ A')} = \frac{(3+2)-(2\times 1)}{(3+2)} = \frac{3}{5}. \quad (3)$$

The same calculus is performed to the output ports as well, generating a proportional factor α_o:

$$\alpha_o = \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}, \quad (4)$$

and the same calculus is performed to multiports as well, generating a proportional factor α_m:

$$\alpha_m = \frac{(0+0)-(2\times 0)}{(0+0)} = 0\,. \tag{5}$$

The average of α_i, α_o, α_m is then multiplied to maximum port penalty, thus generating the port penalty. In the example of Figure 5, the final result of the port comparison between A and A' is:

$$\begin{aligned} PnP_{A,A'} &= \left(\frac{\alpha_i+\alpha_o+\alpha_m}{3}\times 128\right) \\ &\quad + \left(\frac{\frac{3}{5}+\frac{1}{2}+0}{3}\right)\times 128 \\ &\cong 46.86 \end{aligned} \tag{6}$$

Relationship

The fourth comparison element is the set of input relationships (IR) and output relationships (OR) of one activity. Similarly to the penalties applied to ports, the penalty applied to relationship comparison is also proportional to the number of different input and output relationships between a pair of activities. In this paper we consider a relationship as similar or different depending on the activities that it connects. To simplify, let us explain with a theoretical example.

Figure 6 presents a comparison between activities A and A'. A has two output relationships that connects A to B and A to C while A' has two output relationships that connects A' to B' and A' to C. In this paper we consider that an activity has a similar relationship to another if the relationship connects the activity that is being compared to another with the same name. In the example of Figure 6, the relationship that connects A and C is considered similar to the one that connects A' and C because both relationships represent an output relation that is connected to the activity C.

Similarly to the port comparison, in the case of relationship comparison we should calculate proportional factors α_i and α_o to the input and output relationships, respectively. In Figure 6 we may calculate α_o as following:

$$\begin{aligned} \alpha_o &= \frac{(OR\ of\ A + OR\ of\ A') - (2\times matches)}{(OR\ of\ A + OR\ of\ A')} \\ &= \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}\,. \end{aligned} \tag{7}$$

The same calculus is performed to the input relationships as well, generating a proportional factor α_i. In the example of Figure 6, since we are comparing A and A', they do not have input relationships leading to $\alpha_i = 0$.

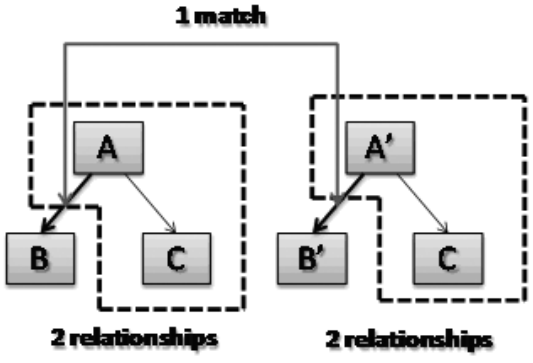

Figure 6 – Relation comparison.

The average of α_i and α_o is then multiplied to the maximum penalty for relationships leading to the relationship penalty. In the example of Figure 6, the final result of the comparison between A and A' is:

$$\begin{aligned} PnR_{A,A'} &= \left(\frac{\alpha_i+\alpha_o}{2}\times 64\right) \\ &= \left(\frac{0+\frac{1}{2}}{2}\right)\times 64 \\ &= 16 \end{aligned} \tag{8}$$

Internal structure – XML definition

This item corresponds one of the ways to evaluate the internal structure of activities. The analysis of the XML definition criteria is used when both activities to be evaluated are not subworkflows (see the next subsection). The comparison element is the XML definition (tags) of each activity. Similarly to the penalties applied to ports and relationships, the penalty applied to XML definition comparison is also proportional to the number of different elements found in the XML definition between a pair of activities. In this paper, we consider an XML element as similar if the string that defines the XML tag is the same and vice-versa. To simplify, let us explain with a theoretical example.

Figure 7 presents a comparison between the XML definition of activities A and A'. A has two elements in the XML (tags *arbitrarywsdl* and *description*) while A' has two elements in the XML as well (tags *arbitrarywsdl* and *description*). In our cartridge implementation, the penalty for the XML definition considers the number of differences found in the strings for each tag of activity A and A'. In the example of Figure 7, one of the two tags is the same in both XML definitions while the other is different.

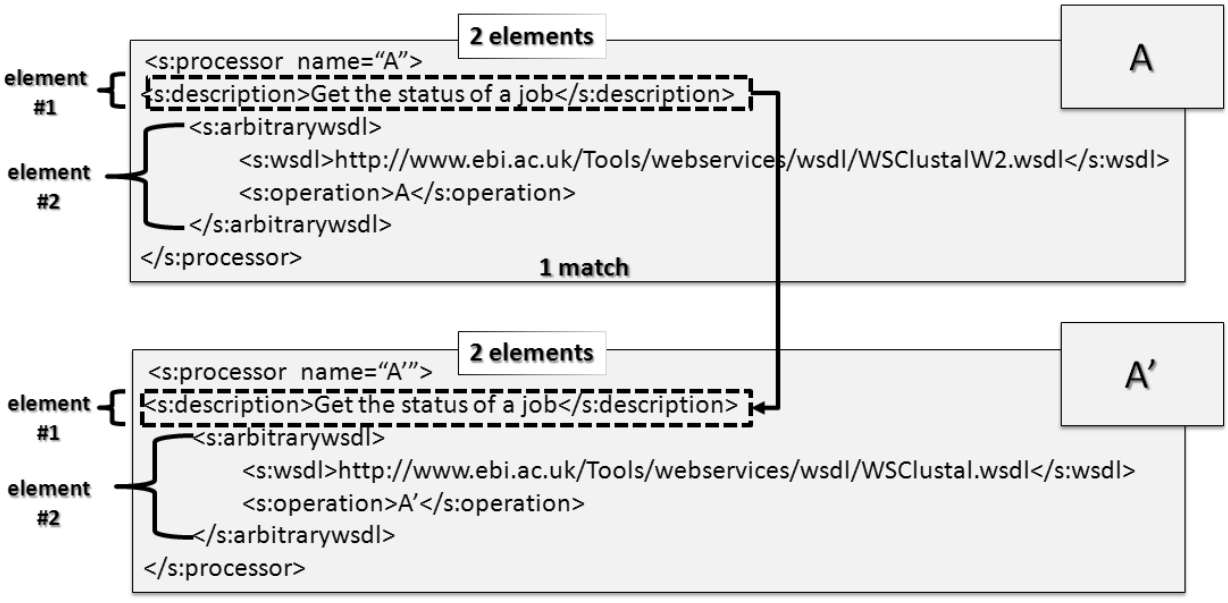

Figure 7 – XML definition comparison.

Similarly to the port and relationship comparisons, in the case of XML definition comparison, we should calculate the proportional factor α. In Figure 7 we may calculate α as follows:

$$\alpha = \frac{(Elem.\ of\ A + Elem.\ of\ A') - (2\ matches)}{(Elem.\ of\ A + Elem.\ of\ A')} = \frac{(2+2)-(2\times 1)}{(2+2)} = \frac{2}{4} = \frac{1}{2}. \quad (9)$$

In this way, the proportional factor α is then multiplied to the maximum penalty, thus generating the real penalty. In the example of Figure 6, the final result for the comparison of A and A' is:

$$PnR_{A,A'} = (\alpha \times 64) = \left(\frac{1}{2} \times 64\right) = 32 \quad (10)$$

Internal structure – subworkflow

The last comparison element is the subworkflow. Similarly to the penalties applied to ports and relationships, the penalty applied to subworkflow comparison is also proportional to the number of differences found between a pair of activities A_i and A_j. In the case of subworkflow comparison we have to separate it in three different cases: (i) one activity is a subworkflow and the other is not, and (ii) both activities are subworkflows. Each one of the cases is detailed in the following using theoretical examples comparing activities A and A'.

One activity is a subworkflow and the other is not. In this first case the algorithm compares activities A and A'. In Figure 8 when it verifies that A is a subworkflow (composed by activities A_1, A_2, and A_3) and A' is a regular activity, it computes the maximum penalty for subworkflow comparison. In Figure 9 when the algorithm verifies that A' is a subworkflow (composed by activities A'_1, A'_2, and A'_3) and A is a regular activity, it computes the maximum penalty for subworkflow comparison. In both cases the algorithm computes $PnSw_{A,A'} = 256$.

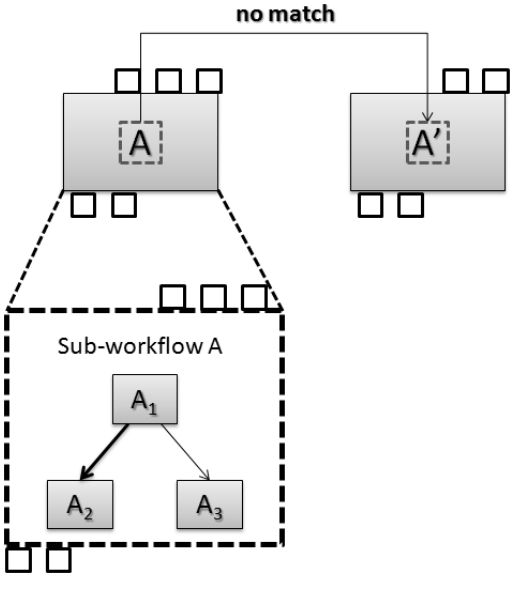

Figure 8 – Subworkflow comparison – first case.

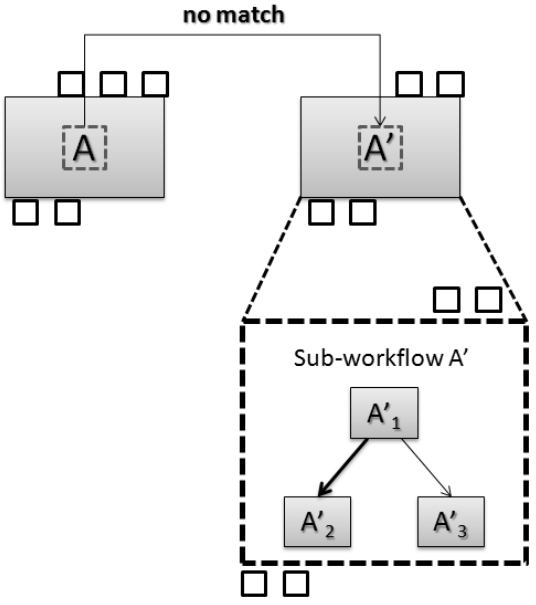

Figure 9 – Subworkflow comparison – second case.

Both activities are subworkflows. This case (Fig. 10) is quite different from the previous two. In Figure 10 both A (composed by activities A_1, A_2 and A_3) and A' (composed by activities A'_1, A'_2 and A'_3) are subworkflows. The algorithm performs a recursively call for each subworkflow. In other words, the whole penalty phase and similarity phase of the algorithm (that is further explained in sub-section 2.3.2) are applied to the subworkflows. The final result is a dissimilarity factor λ (with values varying from 0 to 1) that is multiplied to the maximum penalty in this case. In the example of Figure 10, let us suppose that the dissimilarity factor was calculated as $\lambda = 0.55$. The final penalty in the subworkflow comparison between A and A' is $PnSw_{A,A'} = 128 \times \lambda = 256 \times 0.55 = 140.8$.

2.3.2 Similarity phase

After performing the penalty phase, the algorithm starts the similarity phase. In this phase, once the value of $PnTotal$ for each pair (A_i, A_j) of activities was calculated in the previous phase, the total similarity between W_x and W_y is computed by running a greedy algorithm (Cormen et al. 2001). Initially, this algorithm orders the values of $PnTotal$ ascending to obtain the pairs of activities that are possibly more similar (i.e. which penalties are lower). The pairs are ordered from lower values of $PnTotal$ (possibly the pairs with more similarity) to the pairs with higher values of $PnTotal$ (possibly the pairs with less similarity).

In addition, the algorithm should consider the possibility that an activity A_i of W_x may not be similar to any activity A_j of W_y, and, equivalently, an activity A_j may not be similar to any activity A_i (i.e. activities A_i and A_j are different from all others). So, each activity A_i has a pair $(A_i, \emptyset)$, in which $\emptyset$ denotes an empty activity; similarly, each activity A_j has a pair $(\emptyset, A_j)$. This type of pair means that A_i and A_j are being compared with none of the activities of the opposite workflow, since they are not similar to any of them.

In this way, this pair has the maximum penalty for each element, and the $PnTotal$ is the maximum penalty value, that is 1,984 (sum of the maximum penalty values for each one of the criterion). The pairs $(A_i, \emptyset)$ and $(\emptyset, A_j)$ are included in the ordered list to be used by the greedy algorithm and, as they have the maximum penalty value, they are always placed at the end of the list.

Once the ordered list is computed, the algorithm starts taking the first pair (A_i, A_j) of the list (this pair is composed by activities A_i and A_j from workflows W_x and W_y, respectively). This first pair has the lower value of all $PnTotal$ calculated, denoted here as $PnTotal_1$. The algorithm considers that this pair is the most similar of all in the ordered list, and it does not take into account the other pairs to get to this conclusion, as it is a greedy algorithm. After taking this pair out of the order, as A_i and A_j are already included in this pair, the algorithm excludes from the ordered list all pairs that contain activities A_i and A_j. Then, the algorithm continues its execution by taking the next pair, storing the value of $PnTotal$ and excluding unnecessary pairs until the ordered list is empty.

At the end of the process, the greedy algorithm will have collected the N pairs with the lowest values of $PnTotal$ for each pair of activities. The dissimilarity factor, denoted as λ, is then calculated as follows:

$$\lambda = \frac{PnTotal_1 + PnTotal_2 + \cdots + PntTotal_N}{1{,}984 \times N} \quad (11)$$

The similarity factor, denoted as λ', can be calculated as follows:

$$\lambda' = 1 - \lambda \quad (12)$$

2.4 Clustering

After calculating the similarity between pairs of workflows, it is necessary to cluster them. The cartridge for clustering identifies the similarity of all pairs of workflows that exist in the GExpLine schema and produces a complete graph (Cormen et al. 2001)

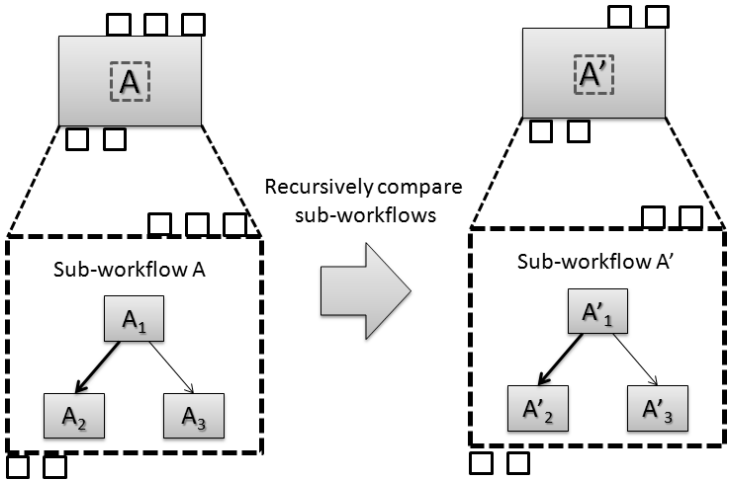

Figure 10 – Subworkflow comparison – third case.

(i.e., a graph in which each pair of vertices has an edge connecting them) named G, with n vertex (n is the total number of workflows stored in the GExpLine schema) and m edges ($m = C_2^n$). After generating this complete graph, SimiFlow prunes the edges that have a similarity value below a threshold. The final result is a forest, which a disjoint union of graphs (Cormen et al. 2001). A set of disconnected graphs (Cormen et al. 2001) is produced and the disconnected graphs composed by one vertex are eliminated. The remaining disconnected graphs represent the clusters of scientific workflows.

The threshold of this clustering process is an input parameter of the algorithm. It determines how much similarity is necessary in order to consider two workflows as part of the same cluster. Choosing the value of the threshold is an *ad-hoc* decision, based on assumptions and prior knowledge and experience. One can assume that two workflows with a similarity value of 0.60 are sufficiently similar to belong to the same cluster, while other can assume that they are totally different. The same issue appears when executing other classic clustering algorithms, like K-means, in which the number K of clusters is an input parameter, and an inappropriate choice of K may yield poor results (Cormen et al. 2001).

Figure 11 presents an example of a cluster. Based on the similarity values, we can note that there are two groups considering a threshold value of 0.55, since all edges with values lower than 0.55 would be pruned. In this way, workflows #1 and #2 are part of one cluster and workflows #3 and #4 are part of another cluster. SimiFlow can be seen as a test bed for such clustering algorithms, because other algorithms can be tried out with low effort since SimiFlow is designed to be extensible.

3 EXPERIMENTAL RESULTS

The SimiFlow architecture was implemented in the GExpLine tool. A first study was conducted to evaluate the viability of the architecture in Silva et al. (2010). An additional experimental study was conducted, reported in this paper, to evaluate the comparison and the clustering algorithms described in the previous sections. The experiment aimed at analyzing the comparison of workflows pairs and identifying clusters of workflows. All workflows used for this study were imported from the *myExperiment* site at December, 2009.

The importing process for this experiment stored a total number of 463 scientific workflows in the GExpLine schema. The total time for importing these workflows was approximately 1 hour and 40 minutes. Considering this scenario, the similarity process would need to perform 106,953 comparisons (C_2^{463}) between pairs of workflows. For an initial study, the computation of similarities would demand too much CPU time to be completed. To minimize the total amount of time to execute the entire experiment, the initial set of concrete workflows was reduced. The experiment used workflows whose name begins with "EBI". After this filter, the number of workflows was reduced to 36 workflows ($C_2^{36} = 630$ comparisons). The machine in which SimiFlow was executed using Hydra middleware (Ogasawara et al. 2009a) is the SGI Altix ICE 8200 with 64 nodes (Intel Xeon 8-core processor). The experiment used four nodes (32 cores). The similarity computation for these 36 workflows (630 comparisons) using Hydra took about one hour.

After the comparisons, the method of clustering was executed and different groups of workflows were created. Table 2

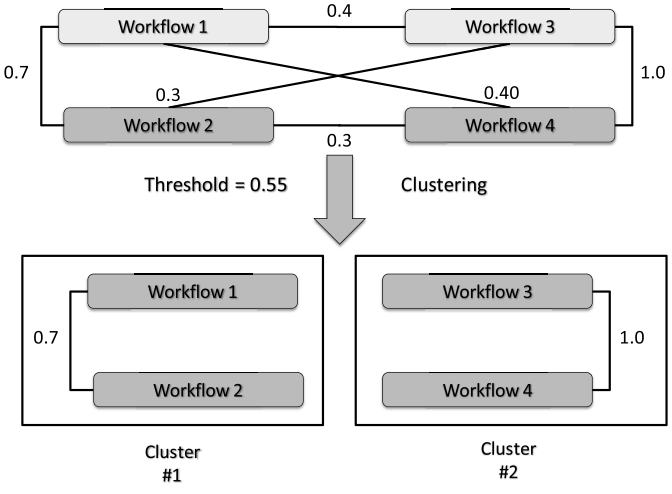

Figure 11 – Example of a workflow cluster.

Table 2 – Workflow clustering.

Workflow name	Cluster number
EBI InterProScan v1	#1
EBI InterProScan v2	#1
EBI_FASTA	#2
EBI_blastpgp_PSI-BLAST	#2
EBI_NCBI_BLAST	#2
EBI FASTA with prompts	#3
EBI_NCBI_BLAST_with_prompts	#3
EBI_ClustalW2_phylogentic_tree	#4
EBI_ClustalW2	#4
EBI_InterProScan_tmhmm_signalp	#5
EBI_InterProScan v3	#5
EBI_ScanPS	#6
EBI_Phobius	#6
EBI_MPsrch	#6
EBI_MaxSprout	#7
EBI_Kalign	#7
EBI_TCoffee	#7
EBI_MUSCLE	#7
EBI_MAFFT	#7

presents the formed clusters and Figure 12 presents the formed clusters in a graph representation. In both Figure 12 and Table 2 each workflow that is isolated (not part of a cluster) is not considered. Based on the similarity values, we may note that there are two groups considering a threshold value of 0.70, since all edges with values lower than 0.70 were pruned. The clusters #1, #2, #3, #4, #5, #6 and #7 were created and the workflows within the same cluster present a similar structure[4].

4 RELATED WORK

There are some approaches in the literature that propose mechanisms to cluster sets of scientific workflows. The approach proposed by Santos et al. (2008) is the most similar to the SimiFlow architecture since it creates clusters of workflows. In this approach workflows are represented using graphs and multidimensional arrays. In addition, it uses the common sub graph metrics (Bunke and Shearer 1998) in the case of graph repre-

[4]The clustered workflows can be viewed at http://gexp.nacad.ufrj.br/news/simiflow

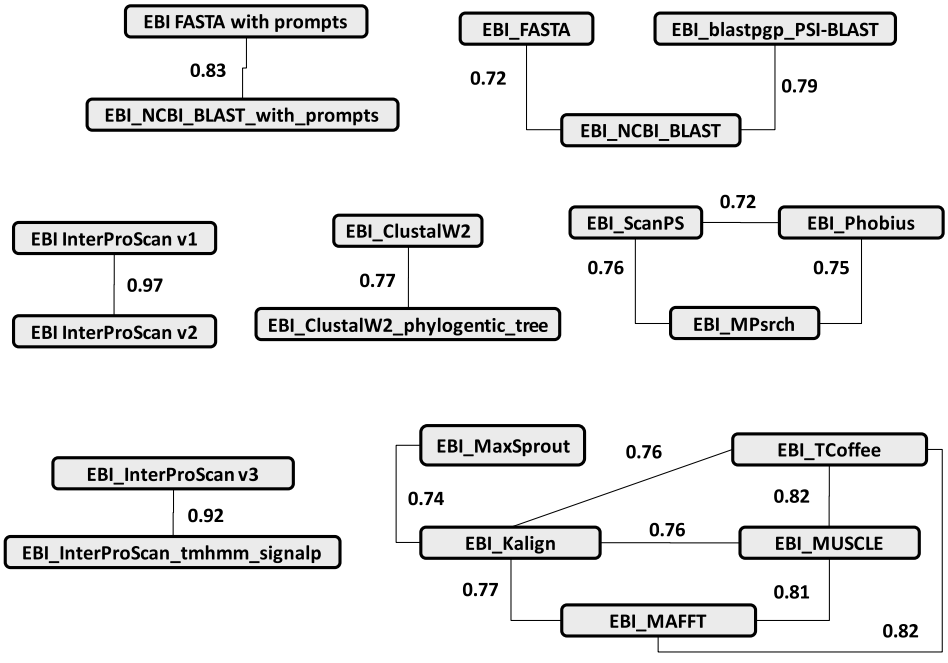

Figure 12 – Representation of workflow clustering.

sentation and cosine distance in the case of multidimensional arrays. The clustering algorithms used were the K-means (Han and Kamber 2006), K-medoids (Han & Kamber 2006) and hierarchical algorithm (Jain et al. 1999). Although the results were satisfactory the workflow representation using graphs or arrays do not provide specific properties of scientific workflows such as input and output ports, for example.

The work proposed by Jung & Bae (2006) proposes a methodology of workflow clustering based on process similarity measures. The main purpose of workflow clustering in this work is to support process repository analysis. Similarly to Santos et al. (2008), Jung & Bae (2006) calculate the similarity of activities of the workflows by using the Cosine measure. Due to that, the algorithm is based on using multidimensional arrays to represent workflows. However, when we represent a workflow using multidimensional arrays we cannot provide specific properties of scientific workflows such as input and output ports, for example.

There are some work that presents mechanisms and techniques to compare data models. Ohst et al. (2003) introduces a method to identify differences between versions of an UML diagram. It proposes methods for detecting and communicating changes in the context of software configuration management (SCM) systems. They consider two broad categories of UML document modifications: Intra-node differences and Structural differences. In both categories, the UML diagram is represented using an acyclic graph in the form of a spanning tree (Cormen et al. 2001). The algorithm compares the different levels of the spanning tree with all others. Using the different levels of a spanning tree allows the comparison to be more precise and robust. The main difference between this approach and SimiFlow is that this approach uses graphs to represent workflows while SimiFlow does not. Actually this is the same problem of the approach proposed by Santos et al. (2008): when graphs are used to represent workflows they do not provide specific properties of scientific workflows such as input and output ports, classes of activities and so on.

Another approach, proposed by Uhrig (2008), compares UML class diagrams with low computational cost. To achieve its goal, this approach compares each class of the diagram with all others, thus creating a bipartite graph. In this graph each edge has an associated value that represents the cost of each pair of classes. Similar to the previous ones, this approach uses graphs to represent workflows and presents the same limitations.

Another similar approach to SimiFlow is the SiDiff (2010) approach. The main goal of SiDiff is to compare data models, thus providing comparison and merge algorithms. The comparison algorithm used by SiDiff is similar to the one proposed by Ohst et al. (2003) and uses spanning trees and level comparison.

The similarity between SimiFlow and SiDiff is that both of them apply penalties to the elements when comparing them.

Another approach for reusing scientific workflows is presented by the myExperiment team in Goderis et al. (2008). In this approach, scientific workflows are grouped to be reused. However, this grouping is based on descriptions of scientific workflows instead of considering workflow structures. In addition, it is focused on workflows specified in Taverna format, while SimiFlow is independent of SWfMS.

CONCLUSIONS

Scientific workflows represent a step forward in the management of scientific experiments based on simulations. As there may be different programs and parameter configurations, these experiments can become increasingly complex, and a representation in different abstract levels is important. However, in the existing systems, there is little support to accomplish this task. Especially when scientists already have specified concrete workflows, it is difficult to know that a group of concrete workflows is, in fact, part of the same experiment. These workflows should be analyzed, compared, and clustered in order to create conceptual representations, helping scientists in their experiments.

In this paper we presented SimiFlow, which is an architecture to support workflow comparison and clustering based on similarity. The comparison and clustering of scientific workflows may facilitate the organization of external repositories of scientific workflows such as myExperiment. SimiFlow provides the necessary infrastructure to perform comparison and clustering based on workflow similarity.

SimiFlow is also an extensible architecture and it is based on the concept of cartridges which makes the implementation of different comparison and clustering mechanisms in SimiFlow easier. Although we used scientific workflows of one domain of knowledge (bioinformatics) in the study presented in this paper, experimental results evidenced that the comparison and the clustering algorithms implemented in SimiFlow are a promising solution to generate clusters of scientific experiments. These clusters can be possibly used in the future to create conceptual representations of scientific experiments, such as experiment lines (Ogasawara et al. 2009b).

As future work, we intent to execute the comparison and clustering of all workflows of the *myExperiment* site using parallelism techniques within Hydra middleware. Another future work includes the development of more comparison and clustering algorithms to enhance SimiFlow.

ACKNOWLEDGMENTS

The authors would like to thank CNPq and CAPES for partially funding this work.

REFERENCES

[1] ALTINTAS I, BERKLEY C, JAEGER E, JONES M, LUDASCHER B & MOCK S. 2004. Kepler: an extensible system for design and execution of scientific workflows. In: SSDBM, p. 423–424, Greece.

[2] BIRSAN D. 2005. On plug-ins and extensible architectures. Queue, 3(2): 40–46.

[3] BUNKE H & SHEARER K. 1998. A graph distance metric based on the maximal common subgraph. Pattern Recogn. Lett., 19(3-4): 255–259.

[4] CALLAHAN SP, FREIRE J, SANTOS E, SCHEIDEGGER CE, SILVA CT & VO HT. 2006. VisTrails: visualization meets data management. In: Proc. SIGMOD 2006, p. 745–747, Chicago, Illinois, USA.

[5] CAVALCANTI MC, TARGINO R, BAIÃO F, RÖSSLE SC, BISCH PM, PIRES PF, CAMPOS MLM & MATTOSO M. 2005. Managing structural genomic workflows using web services. Data & Knowledge Engineering, 53(1): 45–74.

[6] CORMEN TH, CLIFFORD S, LEISERSON CE, RIVEST RL & STEIN C. 2001. Introduction to Algorithms. MIT Press.

[7] DEELMAN E, GANNON D, SHIELDS M & TAYLOR I. 2009. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems, 25(5): 528–540.

[8] GAMMA E, HELM R, JOHNSON R & VLISSIDES JM. 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

[9] GOBLE CA & ROURE DCD. 2007. myExperiment: social networking for workflow-using e-scientists. In: Proceedings of the 2nd workshop on Workflows in support of large-scale science, p. 1–2, Monterey, California, USA.

[10] GODERIS A, DE ROURE D, GOBLE C, BHAGAT J, CRUICKSHANK D, FISHER P, MICHAELIDES D & TANOH F. 2008. Discovering Scientific Workflows: The myExperiment Benchmarks, IEEE Transactions on Automation Science and Engineering.

[11] HAN J & KAMBER M. 2006. Data Mining: Concepts and Techniques. Morgan Kaufmann.

[12] JAIN AK, MURTY MN & FLYNN PJ. 1999. Data clustering: a review. ACM Comput. Surv., 31(3): 264–323.

[13] JUNG J & BAE J. 2006. Workflow Clustering Method Based on Process Similarity. Computational Science and Its Applications – ICCSA 2006, p. 379–389.

[14] MATTOSO M, WERNER C, TRAVASSOS GH, BRAGANHOLO V, MURTA L, OGASAWARA E, OLIVEIRA D, CRUZ SMSD & MARTINHO W. 2010. Towards Supporting the Life Cycle of Large Scale Scientific Experiments. IJBPIM, 5(1): 79–92.

[15] McPHILLIPS T, BOWERS S, ZINN D & LUDÄSCHER B. 2009. Scientific workflow design for mere mortals. Future Generation Computer Systems, 25(5): 541–551 (Maio).

[16] OGASAWARA E, OLIVEIRA D, CHIRIGATI F, BARBOSA CE, ELIAS R, BRAGANHOLO V, COUTINHO A & MATTOSO M. 2009a. Exploring many task computing in scientific workflows. In: MTAGS 09, p. 1–10, Portland, Oregon.

[17] OGASAWARA E, PAULINO C, MURTA L, WERNER C & MATTOSO M. 2009b. Experiment Line: Software Reuse in Scientific Workflows. In: 21th SSDBM, p. 264–272, New Orleans, LA.

[18] OHST D, WELLE M & KELTER U. 2003. Differences between versions of UML diagrams. In: Proc. 9th ESEC, p. 227–236, Helsinki, Finland.

[19] OINN T, ADDIS M, FERRIS J, MARVIN D, SENGER M, GREENWOOD M, CARVER T, GLOVER K & POCOCK MR et al. 2004. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20: 3045–3054.

[20] OLIVEIRA D, OGASAWARA E, CHIRIGATI F, SILVA V, MURTA L & MATTOSO M. 2010. GExpLine: A Tool for Supporting Experiment Composition. In: 3rd International Provenance and Annotation Workshop – IPAW, Troy, NY, USA.

[21] SANTOS E, LINS L, AHRENS JP, FREIRE J & SILVA CT. 2008. A First Study on Clustering Collections of Workflow Graphs. Proc. IPAW 2008, Springer-Verlag, p. 160–173.

[22] SIDIFF. 2010. SiDiff, http://www.sidiff.org.

[23] SILVA V, CHIRIGATI F, MAIA K, OGASAWARA E, OLIVEIRA D, BRAGANHOLO V, MURTA L & MATTOSO M. 2010. SimiFlow: Uma Arquitetura para Agrupamento de Workflows por Similaridade. In: IV e-Science, Belo Horizonte, Minas Gerais, Brazil.

[24] UHRIG S. 2008. Matching class diagrams: with estimated costs towards the exact solution? In: Proc. 2008 CVSM, p. 7–12, Leipzig.