Tosta *et al. Journal of the Brazilian Computer Society* (2015) 21:16
DOI 10.1186/s13173-015-0035-y

Journal of the Brazilian Computer Society
a SpringerOpen Journal

RESEARCH Open Access

Improving workflow design by mining reusable tasks

Frederico E. Tosta[1], Vanessa Braganholo[2*], Leonardo Murta[2] and Marta Mattoso[3]

Abstract

Background: With the increasing popularity of scientific workflow management systems (SWfMS), more and more workflow specifications are becoming available. Such specifications contain precious knowledge that can be reused to produce new workflows. It is a fact that provenance data can help reusing third party code. However, finding the dependencies among programs without the support of a tool is not a trivial activity and, in many cases, becomes a barrier to build more sophisticated models and analysis. Due to the huge number of task versions available and their configuration parameters, this activity is highly error prone and counterproductive.

Methods: In this work, we propose workflow recommender (WR), a recommendation service that aims at suggesting frequent combinations of workflow tasks for reuse. It works similarly to an e-commerce application that applies data mining techniques to help users find items they would like to purchase, predicting a list based on other user's choices.

Results: Our experiments show that our approach is effective both in terms of performance and precision of the results.

Conclusions: The approach is general in the sense that it can be coupled to any SWfMS.

Keywords: Workflows; Sequence mining; Workflow composition; Workflow reuse

Background

Introduction

Scientific experiments are usually composed of pipelined programs, which manipulate large amounts of data. Typically, these experiments are built manually by connecting inputs and outputs of programs, thus producing an execution flow [1]. This execution flow, also known as a *scientific workflow*, can be generically seen as a graph composed of a set of *tasks* that are connected through *ports*. Nodes represent tasks, and edges determine how data flows through tasks. Usually, after executing the workflow, the outputs are analyzed and a new design cycle begins: parameters are changed and tasks are replaced, all with the aim of producing better results. Moreover, scientific experiments frequently use third-party code, services provided by scientific workflow management systems (SWfMS), and proprietary code. Figuring out how this diversity of tasks can be connected in a single workflow is difficult, especially because this design process is ad hoc.

An alternative to help solving this problem is to take advantage of provenance information provided by SWfMS [1]. To figure out how to connect two tasks, the scientist can use provenance queries instead of ad hoc search on the Web. Despite that, reusing a task often involves reusing complementary tasks. Finding dependencies among tasks through provenance queries, without additional support, is not trivial. Due to the huge number of task versions available and their configuration parameters, this activity may become heavily error prone and counterproductive. Even if a powerful workflow provenance support is provided, such as the one offered by Vistrails [2, 3], identifying adequate combinations of tasks can be time-consuming and may involve designing complex queries on provenance databases.

Many existing works use provenance for supporting workflow design. Some of them [4–10] allow similarity search of workflows. However, they do not deal with the problem of suggesting new tasks to the user during

*Correspondence: vanessa@ic.uff.br
[2]IC, Fluminense Federal University, Niterói, RJ, Brazil
Full list of author information is available at the end of the article

© 2015 Tosta et al. **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

workflow design. Other existing works [4, 11–23] propose the use of recommendation systems for scientific workflow design. However, most approaches miss the very sequential nature of workflows. Some recommend only a single task at each step. Others do not consider the previously inserted tasks during recommendation. The ones that recognize the sequential nature of workflows restrict the recommendation to contiguous sequences, being unable to deal with noise (optional tasks). Moreover, there are approaches that do not distinguish accidental (less frequent) sequences from sequences that can be really seen as patterns, providing less useful recommendations. Finally, some approaches suffer from scalability problems when the number of workflows increases.

Our work aims at reducing the burden of scientists by helping them during the experimental design. We started by observing that some sequences of tasks are common to several different workflows within a given domain [11, 14–16, 21]. For example, in a set of scientific experiments from a given domain, task D appears after a specific non-contiguous sequence of tasks $A \rightarrow \ldots \rightarrow B \rightarrow \ldots \rightarrow C$ with a certain frequency f. The idea then is that, when a scientist is modeling a workflow and includes the sequence $A \rightarrow \ldots \rightarrow B \rightarrow \ldots \rightarrow C$, we could suggest the use of D after that sequence. This works similarly to an e-commerce application that applies data mining techniques to help users find items they would like to purchase, predicting a list based on other user's choices. Our goal is to speed up the design of new workflows based on the history of previously designed workflows, hoping that past experiences may help to improve the quality of future workflows.

We introduced a preliminary solution to this problem in a previous work [15]. In that initial solution, a single task is suggested at each step. In this paper, we introduce a novel approach named workflow recommender (WR), which extends those previous ideas by allowing a set of tasks to be suggested to the user. Thus, given a sequence of tasks, we suggest other sequences of tasks that frequently appears after that initial sequence of tasks in the frequent workflows. To evaluate our approach, we compare WR with VisComplete [14], the state-of-the-art recommendation system. The results are promising: the performance of our approach was superior to VisComplete in terms of memory consumption and execution time, especially when large numbers of workflows were involved, without sacrificing the precision of the results.

The remaining of this paper is organized as follows. We present related works in the "Related work" section. Our approach is presented in the "Workflow recommender" section. The "Results and discussion" section discusses our experimental results. Finally, we conclude and discuss future work in the "Conclusions" section.

Related work

The challenges regarding scientific workflow reuse reside in taking advantage of existing workflows to compose new ones. In fact, there has been several works on similarity search of workflows [4–10]. However, these works do not deal with the problem of suggesting new tasks to the user during workflow design.

At the same time, SWfMS are providing more functionalities and additional tasks are becoming available. As a consequence, combining them in a workflow is becoming more complex. SWfMS like Taverna [24], Kepler [25], and VisTrails [2] offer rich graphical interfaces in which previously registered tasks can be dragged to the user's workflow. The search for tasks is limited, since they are usually based on input/output ports as well as user-informed tags. Knowledge of which task can be plugged together is still tacit.

A large set of sample workflows is needed for one to gain experience in configuring the workflow tasks flow [18]. These repositories, such as *myExperiment* [26, 27] and *CrowdLabs* [28], contain hundreds of workflows with different goals. Browsing all of them to gain experience in how to connect workflow tasks is counterproductive and time-consuming. To minimize this problem, recommendation systems for scientific workflows have been proposed [4, 11–23]. We describe them in the remaining of this section.

SmartLink

The main goal of SmartLink [11] is to minimize the existing problems in visual programmable dataflow systems [2, 29]. SmartLink keeps a database that stores information of how workflow tasks are connected (they call these connections *links*). A sequence of links from one task to another is called a *path*. For each link, SmartLink stores an integer value that represents how many times that link has occurred in the workflows. The database is then used to help scientists to build new workflows. The system is able to answer queries such as "How can I connect these two tasks?" and "What can be connected to this task?"

SmartLink's recommendations are based on the task ports. Understanding and choosing tasks is difficult. Choosing a port to be the base for the recommendation is even harder. Furthermore, SmartLink does not take previously inserted tasks into consideration when making a recommendation, which decreases the recommendation quality. In fact, the system considers only the source and target ports to calculate its recommendation.

FlowRecommender

FlowRecommender [16] uses the notion of confidence to recommend workflow tasks. The approach works in two

steps. First, the workflow repository is analyzed so that patterns can be extracted. These patterns are based on the notion of *upstream sub-paths*. An upstream sub-path of a task t of workflow w is any path that occurs before t. As an example, in the workflow $A \rightarrow B \rightarrow C$, the set of upstream sub-paths of C is $\{A, B, A \rightarrow B\}$. Then, the next step is to calculate the confidence of task t for each of its upstream sub-paths p. Confidence is calculated as the probability that t appears given that p has already appeared in the workflow. The confidence values that are greater than a given threshold k are kept and used to find the *influencing upstream sub-path.* The *influencing upstream sub-path* of a task t of a workflow w is the upstream sub-path p with the largest confidence and smallest distance (distance is calculated by counting the number of edges from the first task of p to the last task of w). Then, the influencing upstream sub-paths of all tasks of all workflows in the database are stored in a *pattern table*, which is used in the second phase of the approach.

The second phase is performed online. When a workflow is being constructed, FlowRecommender is activated to suggest new tasks. During the recommendation phase, the last task t of the workflow w is analyzed regarding its output types. Then, a set of *candidate tasks* is retrieved. The candidate tasks are those with input types matching the output types of t. For each of these candidates, the influencing upstream sub-path p is retrieved from the *pattern table*. Then, these paths are compared to the workflow w by using a similarity function that takes into account the sequence and location of tasks in p and w. FlowRecommender finally suggests a set of candidates to complete w, ordered by similarity. When no match is found, the suggestion is based on the input/output types only.

FlowRecommender uses the concept of confidence to calculate the suggestions. We claim that using confidence is not enough. Suppose we have a database with 100 workflows. In these workflows, suppose the sequence $A \rightarrow B$ appears only once. Thus, the confidence of B, given the upstream sub-path A is 100 %. Also, consider that the sequence $X \rightarrow B$ appears 40 times and that $X \rightarrow C$ appears 10 times. Thus, the confidence of B given the upstream sub-path X is 80 %. Despite that the path $A \rightarrow B$ appears only once, it generates a higher confidence than another path that appears 40 times and thus should be more relevant. Using support would solve this issue because the support of $A \rightarrow B$ is 1 % and the support of $X \rightarrow B$ is 40 %. Also, FlowRecommender always recommends a single task at each interaction with the user. To complete a workflow with 30 tasks, it would be necessary to interact 29 times with the user. This would not happen could they recommend entire paths instead of single tasks.

VisComplete

VisComplete [14] is currently the state-of-the-art system for recommending completions to workflows during design. It aims at helping users to design visualizations by considering a collection P of previously created workflows. VisComplete is able to suggest parts of workflows, based on common subgraphs in the collection, to complete workflows that are under development.

The problem of completing workflows is defined as follows. Given a partial graph G, find a set of completions $C(G)$ that reflect the existing structures in a collection of (finalized) graphs. Each completion of G, G_C, is a supergraph of G. To find $C(G)$, VisComplete uses two steps. First, the collection of workflows P is pre-processed to create a path summary P_{path}. P_{path} is a compact representation of P that summarizes relationships between common structures in the collection. Then, given a partial workflow G, P_{path} is used to identify tasks and connections that were used with G in the collection P.

Predictions are based on an anchor task. From this task, the path summary is used to identify nodes that are most likely to follow these paths. VisComplete follows both a top-down and a bottom-up approach, since the direction of the connections may vary. The approach is iterative, so at each insertion of a task, new suggestions are made by using the added task as an anchor.

Since the generation of paths is not based on a data mining technique, VisComplete fails in recommending completions to workflows that have noise or infrequently used tasks. Noise tasks are those that are almost never used. These noise tasks, in our view, do not alter the main goal of the workflow being designed. We advocate that they could be discarded during the recommendation search. To better illustrate the problem, consider a frequent sequence of tasks that is stored in the collection of paths: $A \rightarrow B \rightarrow C \rightarrow D$. Consider also that a workflow is being developed using the following sequence: $A \rightarrow X \rightarrow B$, where X is a noise task and cannot be found in any existing sequence in the collection of paths (e.g., an optional filtering task). Since the sequence $A \rightarrow X \rightarrow B$ does not appear in the collection of paths, VisComplete is not able to recommend the completion $C \rightarrow D$ connected to B in this example.

Approaches based on case-based reasoning

Case-based reasoning (CBR) is a problem-solving technique aimed at easing the reuse of past experience in the form of cases. A case represents a problem situation together with the experiences gained during a problem-solving episode, which includes a solution [23]. In order to use CBR in the context of workflow composition, one needs to adapt workflows into cases. There are several works that deal with this problem [13, 17, 23, 30]. The works of Leake and Kendall-Morwick [13], Chinthaka

et al. [17], and Minor et al. [23] deal with the specific problem of using CBR to aid workflow composition.

Phala [13] uses case-based reasoning to suggest additions to workflows during design. It uses the workflow constructed so far as a query. The results above a similarity threshold are shown to the user as suggestions. The suggestions are mined from retrospective provenance data[1] [1, 31]. (Leake and Kendall-Morwick [13] refer to retrospective provenance as "execution traces," terminology also adopted in other works [32]). The reason for this is that execution traces are simpler than the workflow structure, since they are basically sequential. This requires an additional step at query time, since the query is the workflow specification and not an execution trace. To cope with this difference, non-deterministic control flows are removed from the workflow before it is sent as a query to the case repository (each execution trace is considered a case). The authors claim that it would be useful to use prospective provenance instead of retrospective.

A year later, authors from the same research group proposed a new strategy, also based on case-based reasoning, but now using the workflow structure [17]. In this approach, cases consist of annotated workflows. Scientists need to annotate inputs and outputs of their workflows in order for them to be considered in the recommendation process.

Minor et al. [23] focus on business workflows and allow them to be modified during runtime, which is different from the approaches we discuss in this section. However, it can also be used at design time. They use *Agile Workflow Technology* and a specific modeling language (Cake) to be able to modify the workflow at runtime. Although neither of them is targeted to eScience, some of the ideas could be transposed to scientific workflow composition. In this Agile approach, the workflow is first transformed into a case. Then, the case repository is searched and similar cases are retrieved. The best case is chosen based on a similarity threshold. Then, the next step is to determine the exact place in the target workflow that must be altered (change location). To do so, they use the concept of anchors. After this process, the workflow is modified. A controversial aspect of this work is that the modification may be a delete instead of an insert, which may be counterproductive for the user.

Our previous approach

Our previous approach [15] was conceived to allow incremental recommendation, where, for each selected task, a single following task is suggested. Initially, the relations among tasks are extracted. Each of these relations is stored as a pair (s, t), where s is the source task and t is the target task. For each pair, we also store the ports through which they are connected. When a user adds a new task *t'* to the workflow that is being developed, the system proactively finds the most relevant task that can be connected to *t'*. Moreover, it also indicates how they can be connected. The main limitation of this approach is that it suggests a single task at each step.

There is also some other related approaches that work at a more conceptual level. Some of them focus on workflow reuse by providing composition at a high abstraction level [12, 18, 20, 22], automatically generating the executable workflow. Others generate the workflow from execution trace logs and retrospective provenance [33–35]. Still, others offer a recommendation system that relies on social network analysis [19, 21]. Despite extremely relevant, these approaches act over more abstract representations of workflows or at very low level representations (execution logs and traces) and can be seen as complementary to this and other related works on workflow recommendation presented in this section.

Methods

Workflow recommender

A problem that is shared by related work is that they do not evaluate the task sequence as a whole. We thus introduce a novel approach named WR to overcome the aforementioned problems of the existing approaches in literature. WR contributes with the adoption of a well-known data mining technique to recommend tasks to a workflow under design: sequence mining [36, 37]. The use of a sequence mining algorithm has the following positive aspects: (i) it has been defined to identify and manage noise tasks, providing useful recommendations even in situations where there is no perfect path match in previous workflows; (ii) it is focused on a sequence of events, taking precedence order into account; (iii) it is robust and stable and has been extensively tested by commercial applications of different domains for years; (iv) it has a dozen of different implementations available, which provide equivalent results but differ on performance; and (v) it is fomented by an active research network that can come up with better implementations in the future.

Note that several SWfMS are script-based, such as Swift [38], Pegasus [39], and SciCumulus [40], and thus, no graphical interface is provided for workflow design. However, they all deal with data flows that can be represented as directed acyclic graphs (DAG). Our approach is general in the sense that it does not depend on any system in particular, nor on graphical interfaces. As long as the workflow is a DAG, our approach can be applied. In the remaining of this section, we detail how the classic sequence mining problem was adapted to our context. Moreover, we present an overview of WR and detail its two main phases: preparation phase and query and recommendation phase.

Sequence mining in the context of scientific workflows

Since a workflow is composed of a sequence of tasks, we need to consider these sequences to recommend completions. In data mining, sequence mining is a successful technique for finding frequent sequential events in a dataset. However, to be able to use sequence mining, we first need to evaluate the correspondence of concepts used by the algorithm to the scientific workflow scenario.

We introduce an example to ease the understanding of how sequence mining could be used with scientific workflows. This example is based on the e-commerce domain, since it has successfully adopted sequence mining to recommend purchases to users. Table 1 shows a customer whose ID is 6, the date in which (s)he purchased some items, and the ID of the items (s)he purchased. Additionally, Table 2 shows the transaction sequence of several customers.

Graphically, the transactions of customer 6 can be represented as shown in Fig. 1, where the top item represents the first transaction and the bottom item represents the last transaction. Now, consider a workflow composed of seven tasks, as shown in Fig. 2a. We can decompose this workflow into several paths, as shown in Fig. 2b, each of them representing a possible execution sequence. The *path* between two tasks t_1 and t_2 in a workflow W contains the sequence of tasks that indirectly links t_1 to t_2. As an example, given the workflow presented in Fig. 2a, the path between A and F is $A \rightarrow C \rightarrow D \rightarrow E \rightarrow F$. Each path can be seen as a sequence of tasks, where the top one is executed first and the bottom one is executed last. Formally, the decomposition D of a workflow W into its set of paths is represented as $D(W) = \{P_1, P_2, \ldots, P_n\}$, where $n \geqslant 1$ and P_i is a path. The set of paths $D(W)$ can be used to reconstruct W as $W = \{P_1 \oplus P_2 \oplus \ldots \oplus P_n\}$.

Observing Figs. 1 and 2b, we realize that the shape that represents both structures is identical. As depicted in Table 3, in the e-commerce domain, customers buy items in different moments in time. As a result, it is possible to identify buying sequences that frequently occur. For instance, an intuitive frequent sequence of purchases related to the "The Lord of the Rings" film would be "The Fellowship of the Ring" followed by "The Two Towers" followed by "The Return of the King". This way, after detecting that someone bought "The Fellowship of the Ring" followed by "The Two Towers", a recommendation of "The Return of the King" could naturally emerge. Similarly, in the scientific experiment domain, scientists design workflows containing tasks with dependencies among them. As a result, it is possible to identify task sequences (i.e., paths) that frequently occur.

Table 1 Items bought by a customer [36]

Customer ID	Timestamp	Bought items
6	June 25, 1993	30
6	June 30, 1993	40
6	July 10, 1993	70
6	July 15, 1993	90

Table 2 Customer sequences [36]

Customer ID	Sequence of bought items
1	< (30) (90) >
2	< (10 20) (30) (40 60 70) >
3	< (30 50 70) >
4	< (30) (40 70) (90) >
5	< (90) >
6	< (30) (40) (70) (90) >

By contrasting these concepts from the e-commerce domain and the scientific experiment domain, we could also adapt the data mining problem introduced by Agrawal and Ramakrishnan [36] as follows.

Problem definition

Given a database DB of workflow paths (task sequences), the problem of mining sequential patterns in workflows is to find the maximal paths (i.e., paths that are not contained in any other path) among all paths that have a minimum support. Each such maximal path represents a path pattern.

Overview of WR approach

Figure 3 illustrates the overview of our approach, WR. Since the steps we need to execute are time- and resource-consuming, we decided to split WR in two phases: (1) preparation and (2) query and recommendation.

The *preparation phase* occurs once, before the query and recommendation phase. It is responsible for preprocessing the data from previously designed workflows and for executing the sequence mining algorithm. The *query and recommendation phase* is responsible for processing user queries and recommending workflow paths during workflow design.

Preparation phase

As previously discussed, the preparation phase is responsible for preparing the data from previously designed workflows, allowing the execution of the sequence mining algorithm over them. Figure 2 exemplifies the path extraction activity. The goal of this activity is to extract *maximal paths* (those that are not contained in any other path in the same workflow). WR, as well as the approach proposed by Agrawal and Ramakrishnan [36], only considers the maximal path as input for the data mining algorithm, since it avoids repetitive counting of intermediate paths in the same workflow.

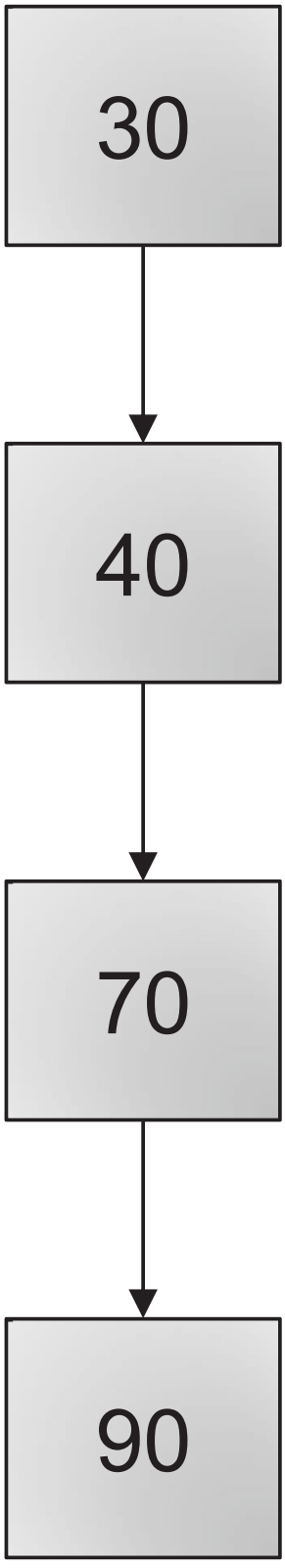

Fig. 1 Graphical representation of a customer transaction sequence

As illustrated in Fig. 4, during the maximal path extraction, paths are segregated according to the domain of the workflow. For instance, paths found in bioinformatics workflows are stored separated from paths found in 3D modeling workflows. This allows us to infer a more precise recommendation because it increases the support of the paths belonging to the domain of the workflow under design. However, this step requires user input to classify the workflows according to their domains.

To exemplify, consider the paths shown in Fig. 5, where path *X*, from the bioinformatics domain, occurs 100 times in the database, i.e., 100 workflows use this path in their composition, and that path *Y*, from the oil domain, occurs 500 times. Consider also that the total number of paths in the database is 1000, from which 200 belong to the bioinformatics domain and 800 to the oil domain. Moreover, suppose a user is developing a new workflow in the bioinformatics domain. When (s)he reaches the stage of development represented by path *Z* in Fig. 5, (s)he asks WR for a recommendation with a minimum support of 20 %. If we consider the complete database (bioinformatics and oil paths), the system would recommend path *Y* with 50 % (500/1000) of support. This is because path *Y* is the only path that contains path *Z* and has a support above 20 %. Despite having the same tasks as path *Z*, path *X* would not be recommended because, in this case, its support would be 10 % (100/1000). Thus, the user would receive an incorrect recommendation to insert task *w* after task *B* in the workflow. However, if we consider only the paths from the bioinformatics domain, the results are completely different. In this case, path *X* would have a support of 50 % (100/200) and would be displayed to the user as a recommendation. Path *Y* would not be taken into account, since it does not belong to the bioinformatics domain. In this case, the next task to be inserted into the workflow would be task *C*.

It is interesting to notice that lower support threshold emphasizes completeness, leading to possible false positives. On the other hand, higher support threshold emphasizes correctness, leading to possible false negatives. This way, the threshold configuration becomes an important task to balance the number and quality of the recommendations.

We also assign weights to the workflows to allow better accuracy in our recommendations. Workflows that are frequently executed may have a higher degree of reliability in the arrangement of their tasks and therefore may contain paths that are more reliable. Those workflows receive a higher weight, leading to higher support in the recommendations of their paths. However, it is worth noting that assigning weights to the workflows can generate side effects: it can dramatically change the support of all paths in the database and, often, exclude paths from the set of paths' pattern. The algorithm we adopted to assign

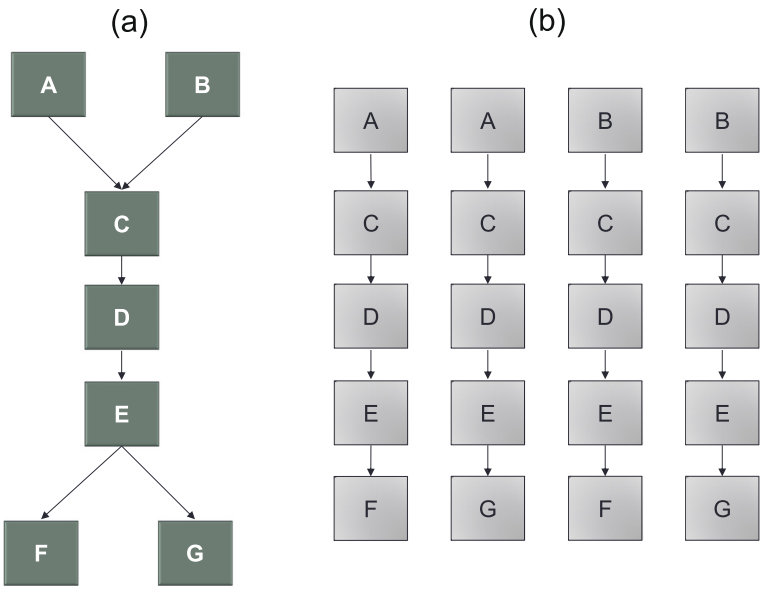

Fig. 2 Workflow (**a**) and its paths (**b**)

weights consists of considering the number of executions of a given workflow. Suppose that a workflow W has been executed n times. Moreover, also suppose that this workflow has a set P of paths $\{P_1, P_2, P_3, \ldots, P_j\}$. When we store these paths in the database, we store each of them n times. This way, the effect is equivalent to a situation where the paths in p occur in n different workflows.

Finally, with all paths extracted from the workflows and inserted into the path database, the next step of the preparation phase is to run the sequential mining algorithm on all databases (complete database and domain-segregated databases). The mining algorithm generates a set of path patterns for each of these databases. This set of path patterns serves as a basis to generate recommendations. In this phase, we also analyze all the paths and generate a list containing all tasks that are present in at least one of the path patterns. We call this list as the *list of minimal sequences*, and we use it in the next phase of our approach. With the preparation phase completed, the next step is the query and recommendation phase.

Query and recommendation phase

As previously discussed, the query and recommendation phase is responsible for receiving user queries and returning the best recommendations, reverse ordered by relevance. The recommendations are nothing more than the most frequent sequences found by the sequence mining algorithm, which in practice are the most common paths found in the workflows. Since the databases are already loaded with the most common paths and the user has chosen which domain (s)he wants to get recommendations for (or has chosen to use the complete database, which contains all paths), the user can start the search process to receive recommendations.

The query and recommendation phase begins every time a user sends a query to the system, as depicted in Fig. 6. This query, handled by step Receive Query (1), is represented by an incomplete path of the workflow, which can contain one or more tasks. In the example of Fig. 6, the query path is $A \rightarrow B$. The next step, Clean Query (2), is responsible for eliminating from this query the tasks that are not present in the list of minimal sequences, i.e., it eliminates tasks that do not appear in any path pattern. The query result is not affected by this removal, since the search for paths that contain the removed tasks would have returned empty anyway. The reduction of the input sequence actually reduces the computational resources needed to execute the next steps. To illustrate the Clear Query (2) step, consider that a user wants to obtain a recommendation for the path $A \rightarrow B \rightarrow C \rightarrow Y$, where Y is a task that is not present in any path pattern. After the Clean Query (2) step, the query becomes $A \rightarrow B \rightarrow C$,

Table 3 E-commerce concepts mapped into scientific workflow concepts

Domain	Concepts			
E-commerce	Customer	Timestamp	Item	Buying sequence
Scientific experiment	Scientists	Dependency	Task	Workflow path

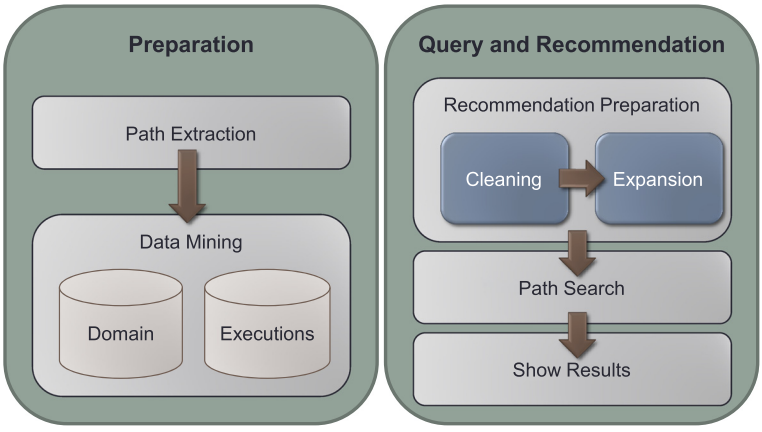

Fig. 3 WR approach

since Y is not present in the list of minimal sequences. This cleaning allows recommending sequences of type $A \rightarrow B \rightarrow C \rightarrow (X)$, where (X) is, in turn, a path containing one or more tasks, to be recommended to the user. This would not be possible if the task Y was present.

The third step of the query and recommendation phase, Expand Query (3), is aimed to find the possible subsequences contained in the query produced in the previous step. To illustrate, consider Fig. 7. Figure 7a shows a possible query, and Fig. 7b shows all subsequences generated from the query. With this procedure, it is possible to generate recommendations for all the branches of a workflow. To exemplify this type of recommendation, consider the query from the previous example. Figure 8a displays the recommendations found by the system to the sub-path $A \rightarrow B$, which is the task Y, and to the sub-path $B \rightarrow C$, which is the task X. Figure 8b shows the workflow after these recommendations are incorporated.

The Expand Query (3) step also favors tasks that have a high degree of synergy, even when they contain noise tasks between them. To illustrate, consider that a user wishes to obtain a recommendation for path $A \rightarrow W \rightarrow B$, where all three tasks are present in the list of minimal sequences, but no maximum path contains these three tasks simultaneously. Consider also that the path $A \rightarrow B \rightarrow C$ has a very high support. If the query were strictly $A \rightarrow W \rightarrow B$, the system would not return any results, but with the execution of Expand Query (3) step, the sub-queries $A \rightarrow W$, $W \rightarrow B$, and $A \rightarrow B$ would be generated, among others, and a recommendation of task C would be generated to the path $A \rightarrow B$, resulting in the path $A \rightarrow W \rightarrow B \rightarrow C$.

Finally, the Search (4) step is responsible for searching the selected databases to find path patterns that contain some of the combinations of paths generated in the previous step. A path P_1 is *contained* in a path P_2 if all tasks in P_1 are in P_2 in the same order, without necessarily being directly connected. For example, $C \rightarrow D$ and $A \rightarrow C \rightarrow D$ are contained in $A \rightarrow B \rightarrow C \rightarrow D$. However, $D \rightarrow A \rightarrow B$ is not contained in $A \rightarrow B \rightarrow C \rightarrow D$. Thus, it is possible to obtain a recommendation for the two ends of the path, i.e., in both directions of the workflow. The final product of this step is a list

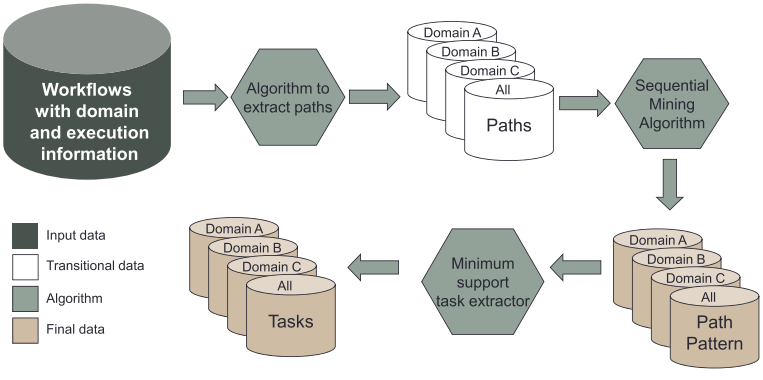

Fig. 4 WR preparation phase overview

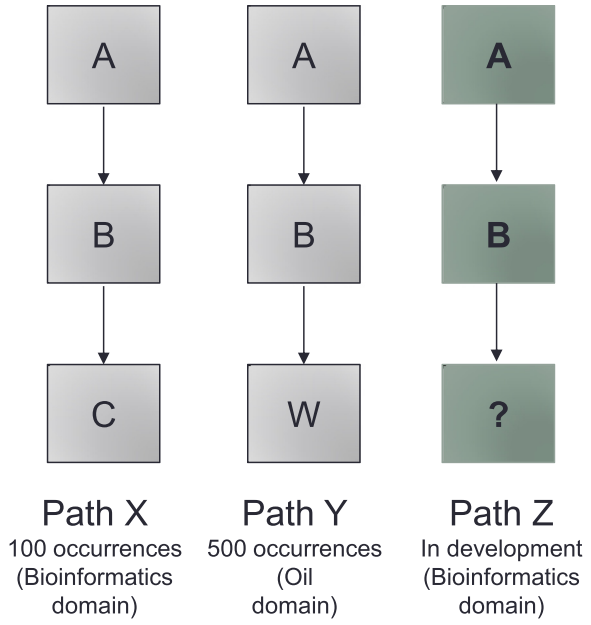

Fig. 5 Paths separated by domains

of path patterns, ordered by relevance and separated by domain. The last stage of the query and recommendation phase, the Recommendation (5) step, consists of showing to the user, in a reverse order, the results obtained in the previous step.

Results and discussion

This section presents an evaluation of the proposed approach. We evaluated both functional and non-functional requirements. Functional requirements are those that describe the behavior of the system and lead to assessing how precise are the recommendations of the proposed approach. On the other hand, non-functional requirements are those that express quality attributes such as reliability, performance, robustness, etc. and lead to assessing how fast is the proposed approach. We adopted VisComplete [14] as a baseline of our experiment, since it is currently the state-of-the-art recommendation system for workflows, with very good recommendation results.

For this evaluation, we used a workflow database gently provided by the creators of VisComplete, which consists of 3343 workflows related to the visualization domain. This database is the same that was used in the experiments with VisComplete [14]. Additionally, we used some examples bundled with the default VisTrails release, which also belong to the visualization domain. As all workflows were related to the visualization domain, it was not necessary to segregate databases per domain during the preparation phase. Moreover, as we did not have information about the number of executions of each workflow, we assumed that all workflows have the same weight.

The evaluation was performed on an Intel Core 2 Quad 2.6-GHz computer, with 3 GB of RAM, and 750 GB of hard drive, running Microsoft Windows 7 Professional x64. We used MySQL 5.1 Community Server as the database. Our prototype was implemented in Python 2.6. We analyzed different algorithms for sequence mining, such as AprioriSOME and AprioriAll [36], GSP [37], WAP [41], and PLWAP [42]. As all of them deal with the same problem definition, they all provided the same results in terms of recommendations, only differing in terms of performance. We adopted PLWAP in WR, since it was the one with the best performance. However, the concrete implementation of the sequence mining algorithm can be replaced with no impact in the prototype. Finally, we used the curve fitting program Fit Lab [43] to analyze the resulting graphs.

The remainder of this section presents the evaluation of the preparation phase, the evaluation of the query and recommendation phase, the evaluation of recommendations

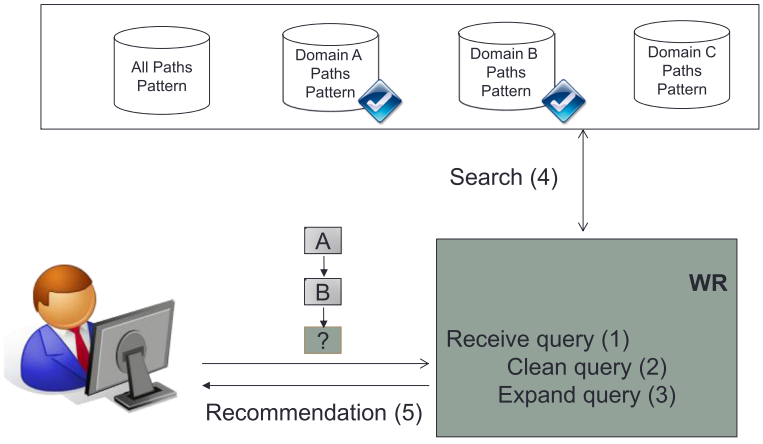

Fig. 6 Query and recommendation phase overview

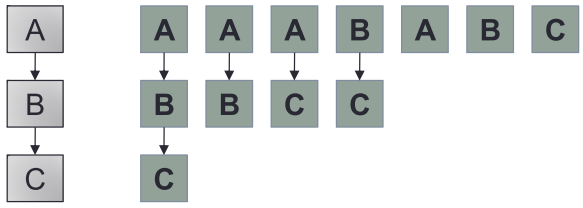

Fig. 7 Query (**a**) and sub-queries (**b**)

in the presence of noise and a brief discussion about some threats to the validity of this experiment.

Preparation phase evaluation

The requirements assessed in the preparation phase are all non-functional. We evaluated both the time required to perform all steps of the preparation phase and the maximum amount of memory used during the preparation process.

Due to difficulties to obtain larger databases to conduct the evaluation of non-functional requirements (performance), it was necessary to replicate the database obtained with the creators of VisComplete, generating larger and smaller databases in terms of the number of workflows. Table 4 displays a summary of the amount of workflows, connections, and paths evaluated. It is important to notice that the original database, provided by the VisComplete team, is database 3, highlighted in italics in Table 4. The database has 3343 registered workflows, with 51,658 connections between tasks. These workflows were used as input to the preparation phase. After the path extraction, 23,623 paths were obtained. The sequence mining identified 2838 pattern paths for a support of 0.5 %. Finally, 65 tasks were extracted from the pattern paths to form the list of minimal sequences. This list and the database of pattern paths were used to evaluate the functional requirements in the query and recommendation phase.

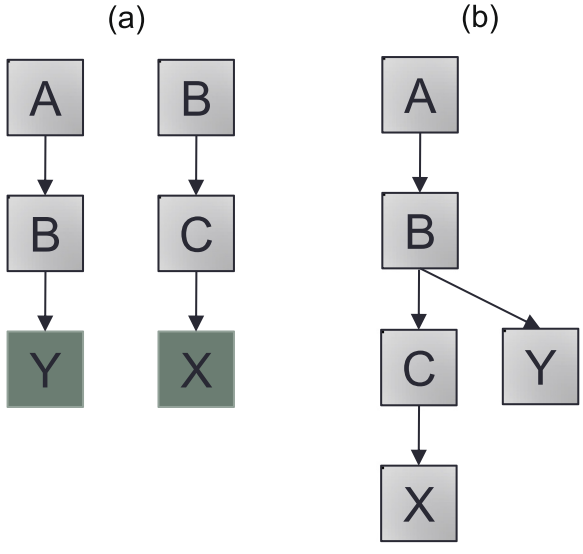

Fig. 8 Recommendations (**a**) and final workflow (**b**)

Table 4 Workflows, connections, and paths per base

Database	Workflows	Connections	Paths
1	835	6,461	2,076
2	1,671	20,321	8,294
3	*3,343*	*51,658*	*23,623*
4	6,686	103,316	47,246
5	13,372	206,632	94,492
6	26,744	413,264	188,984
7	53,488	826,528	377,968
8	80,232	1,239,792	755,936

Table 5 shows the time spent and the memory used to perform the preparation phase of each database for both VisComplete and WR. The support threshold chosen to execute the sequence mining algorithm was 2 %. Figure 9 shows the graphs of the results presented in Table 5.

Analyzing the time graph, it is possible to notice that although the time required to perform the preparation phase was higher in WR, the growth is linear, while in VisComplete, the growth is polynomial (second degree). This was confirmed by the Fit Lab software [43]. On the other hand, the memory graph shows that both the approaches have a linear growth as the number of workflows increases. However, as the slope of the memory used by VisComplete is high, it quickly reached the memory limit, making it impossible to use VisComplete with more than 100,000 workflows in our computer.

The graph analysis shows that WR has advantages in both aspects. In situations where large databases are in place, our approach is still able to process the workflows

Table 5 Time spent and memory usage values for the preparation phase

Database	VisComplete		WR	
	Time (s)	Memory (MB)	Time (s)	Memory (MB)
1	0	10	10	9
2	1	51	35	15
3	4	112	80	30
4	9	204	151	54
5	25	401	301	100
6	74	778	586	193
7	250	1,530	1,245	378
8	803	2,388	1,805	540

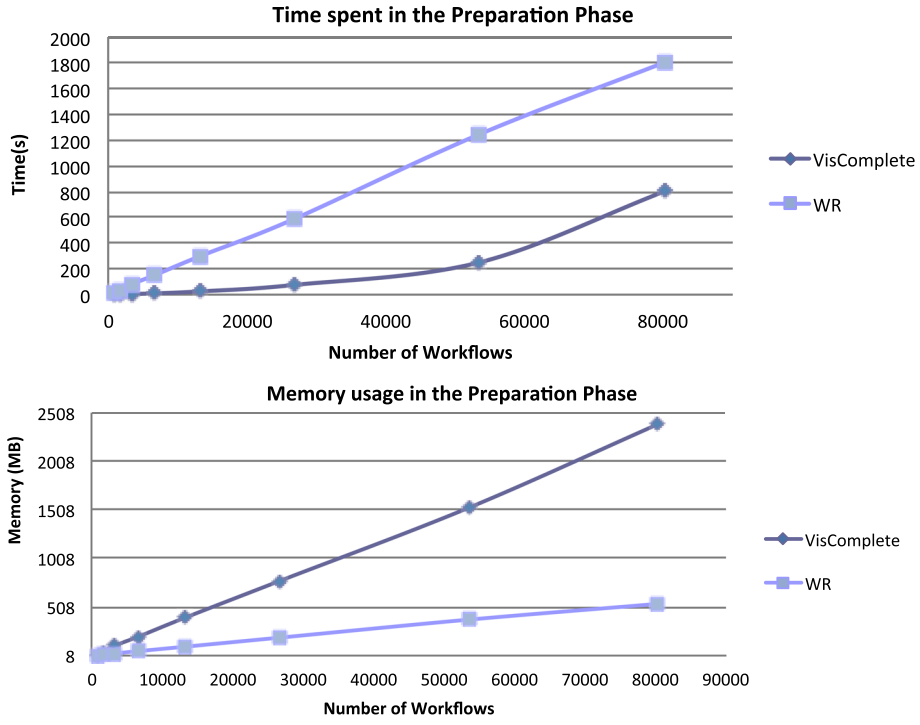

Fig. 9 Time spent and memory usage graphs for the preparation phase

in linear time and with low memory consumption, if compared to VisComplete. Moreover, as previously discussed, it is important to consider that the preparation phase is carried out sporadically and can be performed using idle time. Only the query phase needs to be executed online, for each user demand.

Query and recommendation phase evaluation

The non-functional requirements evaluated in the query and recommendation phase were the time spent and the memory required to load the database of pattern paths and the list of minimal sequences produced by the preparation phase. We used the same databases listed in Table 4 to perform this evaluation. The obtained results are listed in Table 6 and depicted in Fig. 10.

The analysis of the graphs with Lab Fit software shows that WR has a linear growth with a very low slope regarding the time spent, while VisComplete once again has a second-degree polynomial growth.

Regarding the memory usage, VisComplete has a linear consumption in function of the number of workflows, while the memory consumption of our approach is constant for any number of workflows greater than 3343. This probably occurs due to the replication step. As the paths support does not change to all databases derived from database 3, the mining algorithm always produces the same pattern paths. Consequently, the amount of memory to load these pattern paths does not change.

These results show that WR has a significant advantage over VisComplete to load the databases. This occurs mainly because of the reduced number of paths after the execution of the sequence mining algorithm.

Table 7 shows the total number of paths in each database and the number of pattern paths for a support threshold of 2 %. It is possible to observe that the reduction in the number of paths to be analyzed is greater than 90 % for all cases. As previously stated, databases 4, 5, 6, 7, and 8 were generated via the replication of database 3. Due to that, the path support does not change and, hence, the generated pattern paths are the same.

Table 6 Time spent and memory usage values for the query and recommendation phase

Database	VisComplete		WR	
	Time (s)	Memory (MB)	Time (s)	Memory (MB)
1	0.23	117	0.09	6
2	0.72	126	0.17	7
3	1.85	148	0.43	9
4	3.43	184	0.82	9
5	7.50	255	1.61	9
6	18.18	397	3.20	9
7	50.37	681	6.34	9
8	94.00	963	9.50	9

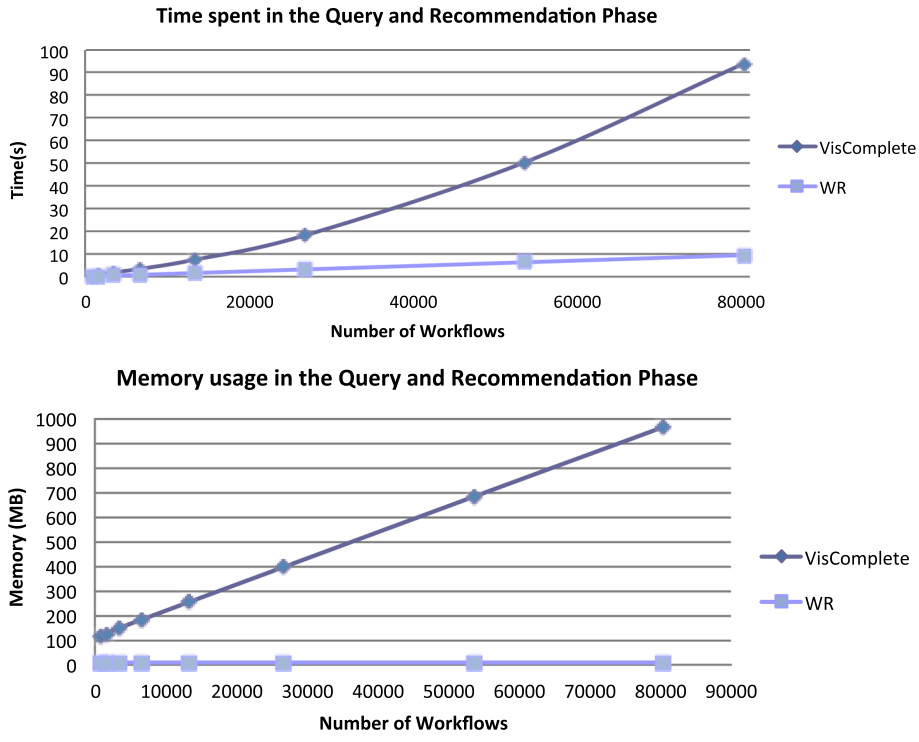

Fig. 10 Time spent and memory usage graphs for the query and recommendation phases

The remaining evaluations in this section are related to the functional requirements of the approach, i.e., the recommendation results. We used only the original database (database 3) in this evaluation for both approaches.

It was necessary to define a dependent variable that indicates the degree of similarity between two paths to allow assessment of the functional requirements of our approach. This similarity function, called *S*, serves for choosing which of the recommendations is the best and is explained in the following.

Consider that someone wants to model the workflow shown in Fig. 11b, which we call intended workflow.

Table 7 Comparison of paths and pattern paths

Database	Paths	Pattern paths	Reduction (%)
1	2,076	169	91.86
2	8,294	282	96.60
3	23,623	1,039	95.60
4	47,246	1,039	97.80
5	94,492	1,039	98.90
6	188,984	1,039	99.45
7	377,968	1,039	99.73
8	755,936	1,039	99.86

When searching for a recommendation after adding the first task, *vtkSphereSource*, the system returned two alternative paths: *A* (Fig. 11a) and *B* (Fig. 11c). Consider that the support of path *B* is higher and, consequently, it is presented first to the user. The support value of a path does not indicate whether it is better or worse, but only if it is more common or not. To determine which of the recommendations is more appropriate, the similarity between the suggested paths and the intended workflow is calculated. The path that obtains the highest degree of similarity is the best recommendation. The similarity *S(q)* is the number of tasks in the recommendation that is exactly in the same sequence in the intended workflow, using task *q* as query. In the example of Fig. 11, path *A* has *S(vtkSphereSource)* = 2 and path *B* has *S(vtkSphereSource)* = 1. In this case, path *A* is the best recommendation and would be selected by the user, automatically adding two tasks in the intended workflow.

To illustrate how the recommendation evaluation was performed for a complete workflow, consider again that someone wants to build the workflow shown in Fig. 11b. By manually adding a first task (*vtkSphereSource*), the system provides two recommendations: *A* and *B*. As the user navigates through the recommendations, it is detected that the recommendation that more closely resembles the workflow of Fig. 11b is the first (path *A*), shown in Fig. 11a, with *S(vtkSphereSource)*

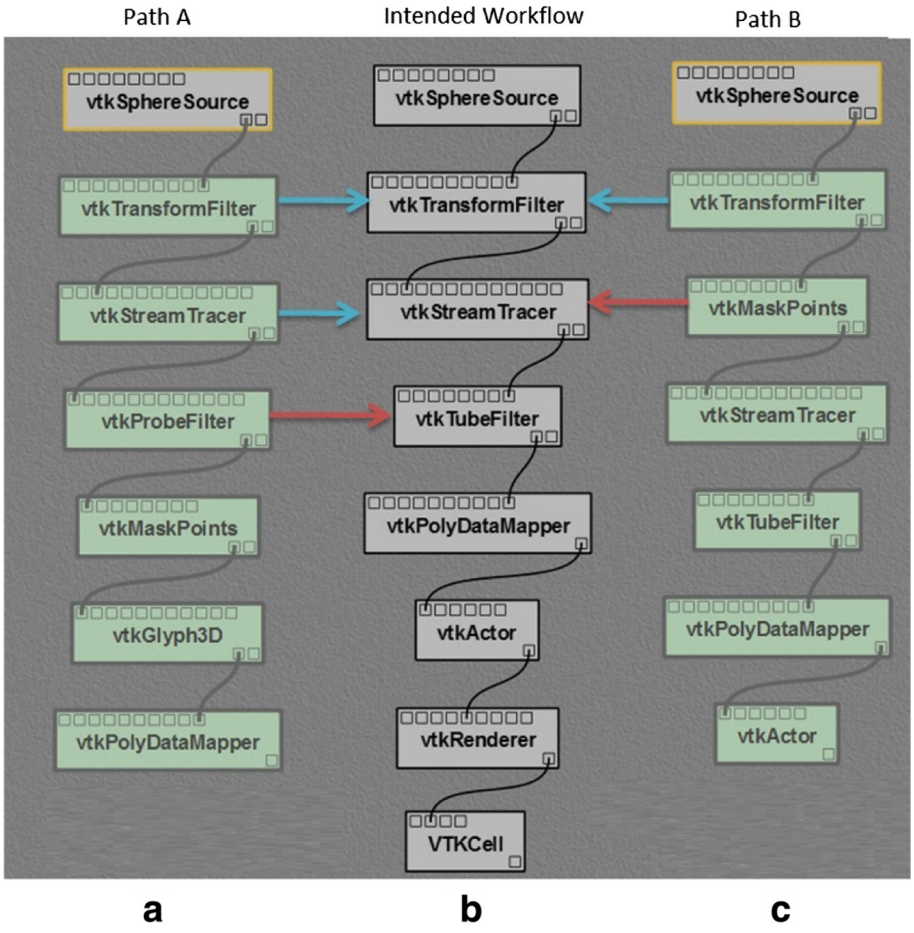

Fig. 11 Similarity among paths. **a** Path *A*. **b** Intended workflow. **c** Path *B*

= 2. The value of the similarity function is also used to prune the recommended path. After accepting the recommendation, all tasks that were within the recommended cutoff point (i.e., *vtkTransformFilter* and *vtkStreamTracer*) are automatically added to the workflow. Subsequently, a second task (*vtkTubeFilter*) is manually added to the workflow and gets no recommendation. By manually adding a third task (*vtkPolyDataMapper*) to the workflow, new recommendations are generated, and the most similar path appears in the fifth position in the list of recommendations with $S(vtkSphereSource \rightarrow vtkTransformFilter \rightarrow vtkStreamTracer \rightarrow vtkTubeFilter \rightarrow vtkPolyDataMapper) = 3$. After automatically adding these three recommended tasks (*vtkActor*, *vtkRenderer*, and *VTKCell*), the workflow is complete.

In the assessment of our approach, we used both the percentage of recommended tasks, *p*, and the ranking average, *a*, to measure the functional capabilities of our approach, also considering VisComplete as baseline. A formal definition of the percentage of recommended tasks is shown in Eq. 1

$$p = \frac{S(q_1) + S(q_2) + \ldots + S(q_n)}{m} \quad (1)$$

In the equation, $S(q_i)$ is the similarity of the best recommendation in the *i*th query, from a total of *n* queries

Table 8 Percentage of recommended tasks and ranking average for VisComplete and our approach

Wf	VisComplete				WR			
	Man	*Aut*	*p* (%)	*a*	*Man*	*Aut*	*p* (%)	*a*
1	11	7	39	1.00	11	7	39	1.00
2	3	5	63	1.80	3	5	63	3.60
3	6	3	33	1.00	6	3	33	1.00
4	8	3	27	1.00	8	3	27	1.00
5	5	8	62	1.25	7	6	46	1.00
6	5	7	58	1.71	5	7	58	1.28
7	6	7	54	3.85	6	7	54	3.42
8	5	5	50	2.40	5	5	50	2.60
Avg.	6.1	5.6	48	1.75	6.3	5.3	46	1.86

that had an accepted recommendation. Moreover, m is the total number of tasks in the intended workflow. According to the previous example, P = (2 + 3) / 8 = 62.5 %. This percentage can be seen as the saved effort during the workflow design. In other words, in this example, the recommendation system automatically modeled 62.5 % (five out of eight tasks) of the workflow. The bigger this percentage, the better is the recommendation system under analysis.

On the other hand, a formal definition of the average ranking a is given by Eq. 2, where $r(q_i)$ is the ranking of the best recommendation in the ith query, from a total of n queries that had an accepted recommendation. To derive this formula, we considered the ranking of the first task of the best recommendation in the ith query as $r(q_i)$ and the ranking of the remaining $(S(q_i) - 1)$ tasks as 1, for a total of $S(q_i)$ recommended tasks. This way, recommendations with the same ranking but higher similarity have a lower a. This reflects the fact that all matched tasks, except the first one, are directly incorporated in the workflow without the need of navigating through more recommendations. According to the previous example, $a = (1+(2-1)+5+(3-1)/(2+3) = 1.8$. This value represents the most probable ranking of each recommended task. The best value for the average ranking is 1, and it only occurs if the best recommendation is always the first one. The lower this value, the better is the recommendation system under analysis.

$$a = \frac{r(q_1) + (S(q_1) - 1) + \ldots + r(q_n) + (S(q_n) - 1)}{S(q_1) + S(q_2) + \ldots + S(q_n)} \quad (2)$$

We randomly chose eight workflows from the database provided by the VisComplete team for this experiment. These workflows played the role of intended workflows, and both VisComplete and WR were queried as each task was added to the workflow, according to the previously mentioned procedure. The results of this experiment are shown in Table 8, where *Wf* represents each of the eight workflows, *Man* represents the manually added tasks, *Aut* represents the automatically added tasks, and p and a are the aforementioned metrics.

The results of Table 8 show that both approaches had similar percentage of recommended tasks, reducing in almost 50 % the necessity of manually adding tasks when modeling workflows, with a peak of 63 %. Even in the worst case, both automatically added 27 % of the workflow tasks. Actually, both approaches had the same functional behavior except in workflow 5, with a better percentage to VisComplete. Likewise, the average ranking shows that the best recommendation is usually the first or second in the ranking for both approaches. The worst case, 3.85, was obtained by VisComplete in workflow 7, and the best case was obtained three times by VisComplete (workflows 1, 3, and 4) and four times by WR (workflows 1, 3, 4, and 5). It is important to notice that these results were obtained over the workflow database provided by VisComplete. In this database, we were not able to explore the capability of handling noise tasks.

Recommendations in the presence of noise

We also wanted to evaluate how both approaches deal with noise tasks in the workflow. To do so, we selected a workflow with optional tasks. It has a task called *vtkStripper*, which does not occur in any other workflow in our database and thus does not appear in the path pattern database.

Our experiment consisted of using part of this workflow and asking for a recommendation. We used the sub-path of the workflow starting from the first task until the *vtkStripper* task (that is, *vtkStripper* was the last task added to the workflow before we ask for a recommendation). Since VisComplete uses the complete query to make a recommendation, and *vtkStripper* is not on any other workflow, the correct suggestion (*vtkProbeFilter*) appears only in the eighth position in the list of suggestions provided by VisComplete.

This kind of problem does not occur in WR, since the *Clean Query* phase removes the noise and searches for only possible paths. In fact, when we manually eliminate the noise and give the workflow to VisComplete, it suggests *vtkProbeFilter* as the first option, just like in our approach. The automatic noise removal provided by our approach is an important advantage if compared to its competitors.

Threats to validity

As every experimental evaluation, our evaluation suffers from some threats to validity. Regarding the non-functional evaluation, it was necessary to replicate the workflow database. However, this replication may have benefited the results of our approach due to the nature of sequence mining algorithms, especially regarding the time and memory analysis in the query and recommendation phases, as discussed in the "Query and recommendation phase evaluation" section.

Moreover, we did not evaluate the usability of our approach compared to VisComplete. VisComplete has a graphical user interface that facilitates the navigation among the provided recommendations and the selection of a specific recommendation. The current version of our prototype has no graphical user interface because usability was not in the scope of our work. However, we believe that our approach could be easily adapted to work under the hood, using the VisComplete graphical user interface as frontend.

Finally, the small number of workflows analyzed in the functional evaluation (eight) and the domain specificity of

the workflow database (visualization) do not allow us to generalize the evidences presented in this section. In addition, some important features of our approach, such as domain segregation and workflow weighing were disabled in this evaluation due to the characteristics of the available workflow database.

Conclusions

Designing workflows is becoming more and more complex, due to the large amounts of services, third-party code, and versions of programs widely available. Connecting such heterogeneous tasks in a single workflow may be error prone and time-consuming. Currently, there is a strong dependency in the individual skills of scientists to connect tasks and design a workflow.

Our work proposes a recommendation service that suggests frequent combinations of tasks, thus accelerating the workflow design. Our approach is the first one to use sequence mining techniques in workflows. Sequence mining is order preserving and overcomes the limitation of related work while examining neighbors rather than the whole dependency task definition. This kind of mining algorithm has been largely applied in e-commerce applications, presenting good results. Our approach is divided into two phases: (1) preparation and (2) query and recommendation. In the preparation phase, we extract the workflow paths separated by application domains, set weights to the mostly used workflows, and apply sequence mining. In the query and recommendation phase, we receive user queries, process them, and return recommendations.

We have implemented a prototype to evaluate our approach. Through experiments, we have compared our approach with VisComplete [14], the state-of-the-art system up until now. Our evaluation has focused on the performance (processing time and memory consumption) of both phases and in the precision of the query and recommendation phase. The results are promising. The performance of our approach was superior to VisComplete, especially when large numbers of workflows were involved. Regarding precision, both approaches presented very close results: the effort of constructing a new workflow was reduced in almost 50 %.

In future work, we intend to build a graphical user interface to our prototype and use this interface to evaluate the usability of our approach. We also plan to evaluate WR with workflows from different domains, so we can measure how workflow segregation by domain, done in the preparation phase, can affect the results. Moreover, we plan to conduct a study with scientists to measure the quality of the recommendations, as well as the time needed to complete the workflow, with and without WR. Finally, since designing a workflow is a trial and error procedure, WR can also be coupled to redesign tools like [44].

Endnote

[1]Retrospective provenance describes the steps taken during execution, while prospective provenance describes the workflow structure [1, 31].

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
MM, LM, and FT participated in the conception of the approach. FT implemented the approach and run the experiments. VB helped in analyzing the final results and writing and polishing the paper. All authors read and approved the final manuscript.

Acknowledgements
We would like to thank the authors of VisComplete and the VisTrails team for kindly providing us both their database and source code for our experiments. This work was partially funded by CNPq and FAPERJ.

Author details
[1]Brazilian Army, Rio de Janeiro, RJ, Brazil. [2]IC, Fluminense Federal University, Niterói, RJ, Brazil. [3]COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil.

Received: 22 October 2014 Accepted: 29 December 2014
Published online: 05 October 2015

References

1. Davidson SB, Freire J. Provenance and scientific workflows: challenges and opportunities. In: Proceedings of the International Conference on Management of Data (SIGMOD). New York, NY, USA: ACM; 2008. p. 1345–1350. June 2008.
2. Callahan SP, Freire J, Santos E, Scheidegger CE, Silva CT, Vo HT. VisTrails: visualization meets data management. In: Proceedings of the International Conference on Management of Data (SIGMOD). Chicago, IL, USA: ACM; 2006. p. 745–747. June 2006.
3. Scheidegger C, Koop D, Santos E, Vo H, Callahan S, Freire J, et al. Tackling the provenance challenge one layer at a time. Concurr Comput Prac Exp. 2008;20(5):473–483.
4. Goderis A, Li P, Goble C. Workflow discovery: the problem, a case study from e-Science and a graph-based solution. In: Proceedings of the International Conference on Web Services (ICWS). Chicago, USA; 2006. p. 312–319. Sept 2006.
5. Santos E, Lins L, Ahrens JP, Freire J, Silva CT. A first study on clustering collections of workflow graphs. In: Proceedings of the International Provenance and Annotation Workshop (IPAW). Berlin, Heidelberg; 2008. p. 160–173.
6. Friesen N, Ruping S. Workflow analysis using graph kernels. In: Proceedings of the ECML/PKDD Workshop on Third-Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD). Barcelona, Spain; 2010. p. 1–12. Sept 2010.
7. Stoyanovich J, Taskar B, Davidson S. Exploring repositories of scientific workflows. In: Proceedings of the International Workshop on Workflow Approaches to New Data-centric Science (WANDS). New York, NY, USA: ACM; 2010. p. 7–1710. June 2010.
8. Silva V, Chirigati F, Maia K, Ogasawara E, Oliveira D, Braganholo V, et al. Similarity-based workflow clustering. J Comput Interdiscip Sci. 2011;2(1): 23–35. doi:10.6062/jcis.2011.02.01.0029.
9. Costa F, Oliveira D, Ogasawara E, Lima A, Mattoso M. Athena: text mining based discovery of scientific workflows in disperse repositories. In: Proceedings of the International Workshop on Resource Discovery. Berlin, Heidelberg: Springer; 2012. p. 104–121. Nov 2010.
10. Bergmann R, Gil Y. Similarity assessment and efficient retrieval of semantic workflows. Inform Syst. 2014;40:115–127.
11. Telea A, van Wijk J. SmartLink: an agent for supporting dataflow application construction. In: Proceedings of the Eurographics and IEEE TCVG Symposium on Visualization. Amsterdam, The Netherlands; 2000. p. 189–198. May 2000.

12. Xiang X, Madey G. Improving the reuse of scientific workflows and their by-products. In: Proceedings of the IEEE International Conference on Web Services (ICWS). Salt Lake City, USA; 2007. p. 792–799. July 2007.
13. Leake D, Kendall-Morwick J. Towards case-based support for e-Science workflow generation by mining provenance. In: Proceedings of the European Conference on Advances in Case-Based Reasoning (ECCBR). Berlin, Heidelberg: Springer; 2008. p. 269–283. Sept 2008.
14. Koop D, Scheidegger C, Callahan S, Freire J, Silva C. VisComplete: automating suggestions for visualization pipelines. IEEE Trans Vis Comput Graph. 2008;14(6):1691–1698.
15. Oliveira F, Murta L, Werner C, Mattoso M. Using provenance to improve workflow design. In: Proceedings of the International Provenance and Annotation Workshop (IPAW). Salt Lake City, USA; 2008. p. 136–143. June 2008.
16. Zhang J, Liu Q, Kai X. FlowRecommender: a workflow recommendation technique for process provenance. In: Proceedings of the Australasian Data Mining Conference (AusDM). Melbourne, Australia; 2009. p. 1–7. Dec 2009.
17. Chinthaka E, Ekanayake J, Leake D, Plale B. CBR based workflow composition assistant. In: Proceedings of the Congress on Services (SERVICES). Washington, DC, USA: IEEE Computer Society; 2009. p. 352–355. July 2009.
18. Mattoso M, Werner C, Travassos GH, Braganholo V, Murta L, Ogasawara E, et al. Towards supporting the life cycle of large-scale scientific experiments. Int J Bus Process Integr Manag. 2010;5(1):79–92.
19. Tan W, Zhang J, Foster I. Network analysis of scientific workflows: a gateway to reuse. IEEE Comput. 2010;43(9):54–61.
20. Oliveira D, Ogasawara E, Seabra F, Silva V, Murta L, Mattoso M. GExpLine: a tool for supporting experiment composition. In: Proceedings of the Provenance and Annotation of Data and Processes. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2010. p. 251–259. June 2010.
21. Zhang J, Tan W, Alexander J, Foster I, Madduri R. Recommend-as-you-go: a novel approach supporting services-oriented scientific workflow reuse. In: IEEE International Conference on Services Computing (SCC). Washington, DC, USA: IEEE Computer Society; 2011. p. 48–55. June 2011.
22. Cerezo N, Montagnat J. Scientific workflows reuse through conceptual workflows. In: Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS). Seattle, USA: ACM; 2011. p. 1–10. Nov 2011.
23. Minor M, Bergmann R, Görg S. Case-based adaptation of workflows. Inform Syst. 2014;40:142–152.
24. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, et al. Taverna: a tool for building and running workflows of services. Nucleic Acids Res. 2006;34(2):729–732.
25. Altintas I, Berkley C, Jaeger E, Jones M, Ludascher B, Mock S. Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings of the Scientific and Statistical Database Management (SSDBM). Greece; 2004. p. 423–424. June 2004.
26. Goble CA, Bhagat J, Aleksejevs S, Cruickshank D, Michaelides D, Newman D, et al. myExperiment: a repository and social network for the sharing of bioinformatics workflows. Nucleic Acids Res. 2010;38(Web Server Issue):677–682.
27. Goble CA, Roure DCD. myExperiment: social networking for workflow-using e-scientists. In: Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS). Monterey, CA, USA: ACM; 2007. p. 1–2. June 2007.
28. Mates P, Santos E, Freire J, Silva CT. CrowdLabs: social analysis and visualization for the sciences. In: Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM). Berlin, Heidelberg: Springer; 2011. p. 555–564. July 2011.
29. Upson C, Faulhaber Jr T, Kamins D, Laidlaw DH, Schlegel D, Vroom J, et al. The application visualization system: a computational environment for scientific visualization. IEEE Comput Graph Appl. 1989;9(4):30–42.
30. Minor M, Bergmann R, Görg S, Walter K. Towards case-based adaptation of workflows In: Bichindaritz I, Montani S, editors. Case-based reasoning research and development. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2010. p. 421–435. July 2010.
31. Freire J, Koop D, Santos E, Silva CT. Provenance for computational tasks: a survey. Comput Sci Eng. 2008;10(3):11–21.
32. Murta L, Braganholo V, Chirigati F, Koop D, Freire J. noWorkflow: capturing and analyzing provenance of scripts. In: Proceedings of the International Provenance and Annotation Workshop (IPAW). Cologne, Germany; 2014. p. 1–12. June 2014.
33. Yaman F, Oates T, Burstein MH. A context driven approach for workflow mining In: Boutilier C, editor. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). Pasadena, USA; 2009. p. 1798–1803. July 2009.
34. van der Aalst W, Weijters T, Maruster L. Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng. 2004;16(9):1128–1142.
35. Zeng R, He X, van der Aalst WMP. A method to mine workflows from provenance for assisting scientific workflow composition. In: Proceedings of the IEEE World Congress on Services (SERVICES). Washington, DC, USA: IEEE Computer Society; 2011. p. 169–175. July 2011.
36. Agrawal R, Ramakrishnan S. Mining sequential patterns. In: Proceedings of the International Conference on Data Engineering (ICDE). Taiwan; 1995. p. 3–14. March 1995.
37. Srikant R, Agrawal R. Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the International Conference on Extending Database Technology (EDBT). Berlin, Heidelberg: Springer; 1996. p. 3–17. March 1996.
38. Wilde M, Hategan M, Wozniak J, Clifford B, Katz D, Foster I. Swift: a language for distributed parallel scripting. Parallel Comput. 2011;37(9):633–652.
39. Deelman E, Singh G, Su M-H, Blythe J, Gil Y, Kesselman C, et al. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. J Sci Program. 2005;13(3):219–237.
40. de Oliveira D, Ogasawara ES, Baião FA, Mattoso M. SciCumulus: a lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In: IEEE International Conference on Cloud Computing, CLOUD 2010. Miami, FL, USA: IEEE; 2010. p. 378–385. doi:10.1109/CLOUD.2010.64. http://dx.doi.org/10.1109/CLOUD.2010.64. July 2010.
41. Pei J, Han J, Mortazavi-Asl B, Zhu H. Mining access patterns efficiently from web logs. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). London, UK: Springer; 2000. p. 396–407. June 2010.
42. Ezeife CI, Lu Y. Mining web log sequential patterns with position coded pre-order linked WAP-Tree. Data Min Knowl Disc. 2005;10(1):5–38.
43. Silva WPd, Silva CM, Silva DD, Soares IB, Oliveira JA, Silva CD. LAB fit curve fitting: a software in portuguese for treatment of experimental data. Revista Brasileira de Ensino de Física. 2004;26(4):419–427.
44. Santos ID, Dias J, Oliveira DD, Ogasawara E, Ocaña K, Mattoso M. Runtime dynamic structural changes of scientific workflows in clouds. In: Proceedings of the International Conference on Utility and Cloud Computing (CloudAM). Washington, DC, USA: IEEE Computer Society; 2013. p. 417–422. Dec 2013.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Immediate publication on acceptance
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com