CBR Based Workflow Composition Assistant

Eran Chinthaka, Jaliya Ekanayake, David Leake, Beth Plale
School of Informatics, Indiana University
Bloomington, Indiana, USA.
{echintha, jekanaya, leake, plale}@cs.indiana.edu

***Abstract*—Composing a scientific workflow from scratch may be time-consuming, even if the scientist is fully aware of the semantics, the inputs, and the outputs of the expected workflow. Reusing existing services and parts from already composed workflows can aid in reducing the total workflow composition time. However, matching the semantics and the inputs and outputs of these reusable components manually is not an easy task, especially when there are hundreds of such components available. Even components are annotated with information on the semantics of their inputs and outputs, the complex nature of the semantic languages may make manual component selection even harder. In this paper, we propose a Case-Based Reasoning (CBR) approach to assist composition of workflows based on the characteristics of the inputs and the outputs of the reusable workflow components, facilitating user exploitation of existing services and workflows during workflow composition. The architecture can also be extended to utilize the semantics of the various components improving the *precision* of the identified reusable components.**

I. Introduction

Scientists compose workflows to achieve a custom task, primarily by aggregating existing services. The outcome can be a final workflow or a part of a much larger workflow. Scientists start the composition by keeping in mind a certain description of the inputs and outputs of the workflow. They then must develop a path from the inputs to the outputs, preferably using the available services and components. Even though this task might appear trivial to perform, the composition might be difficult for various reasons:

- The scientist may have to choose from many available services, making a manual search time-consuming due to the number of candidates.
- Selecting the right service requires sufficient knowledge to understand the semantics of each service to get the best workflow.
- Scientists might also have difficulties expressing their service requirements to the system. Even though various semantic languages [1], [2], [3] have been proposed to describe requirements, some users might not be conversant with those languages.
- Services might have been written using various service description standards with which the scientist is unfamiliar. For example, WSDL (Web services Description Language) [4], which is commonly used to define Web services, might not be familiar to the user.

*This material is based on work supported in part by the National Science Foundation under Grant No. OCI-0721674.

Automated support for identifying and reusing existing services and parts of workflows from the scientists' previous work or from the work of her colleagues or from a public source such as myexperiment.org is an appealing strategy for reducing the development time. These existing workflows or sub-components within workflows may either help the user to accomplish the whole task or might help to achieve sub-parts of his larger workflow.

Even if scientists are not familiar with semantic languages, they should at least be able to express their requirements using simple keywords. Consequently, it is desirable for the system to support retrieval using keyword-based descriptions. However, if the user is advanced enough, then the system should also support expressing the requirements using a given semantic language. We propose a Case-based reasoning based framework to assist users during their workflow composition. Case-based reasoning systems reason from specific prior experiences, retrieving records of similar past problems and adapting them to fit new needs [5], [6].

Our system, now under development, helps users to find existing workflows to fit the users' requirements or to find components within existing workflows which might fit within the user's bigger workflow. It will also enable users to express their requirements in a user-friendly way. In our implementation we used a simple keywords-based approach; however, our framework can be easily extended to be used with the other known semantic description languages.

In this paper, Section II discusses the related work in this domain. Section III explains the proposed solution to overcome the problems outlined above. In Section IV we will discuss the key features of our system. Section V we will outline the future work that can be carried out to improve our proposed system and finally Section VI concludes the discussion.

II. Related Work

The most common method used to help users to compose workflows is to take a set of Web services and automatically compose a set of candidate workflows from it based on semantic service descriptions. Previous approaches have used logic programming [7], type matching [8], linear logic [9], and AI planning [10]. Many of these systems generate a set of workflows, based not on user goals but on the matching of inputs and outputs. Ambite and Kapoor [11] have also proposed a system to automatically compose workflows from a given set of requirements, and have used shim services to bridge between incompatible services. Patrick et al. [12] have proposed a constrained object model for workflow composition, based upon a meta-model for workflows and ontologies for processes and data flows. These approaches use various search strategies to automatically generate a workflow for the user.

Automatic composition of workflows from scratch faces some substantial difficulties. First, when the search space or the number of services increases, generating a solution might be time-consuming, especially for generative search-based approaches. In addition, the required time can increase significantly with the size of the search space, making scale-up an important concern.

Another class of issues concerns usability: Being able to handle different ways of expressing users' inputs and supporting the interactive workflow building process. Users build workflows incrementally and the system must be able to help the users in each stage. Coming up with multiple suggestions prioritized by confidence may be useful for providing useful support with flexibility.

Kim et al [13] discuss CAT, an approach to help users during workflow composition by analyzing the composed workflows. The CAT system notifies users on various errors and suggests the users to overcome them. They have used planning techniques for the better construction workflows. Leake and Morwick [14] propose a case-based approach to proactive retrieval as the background to an incremental workflow construction process, based on analysis of the workflow under construction, but without using explicit goal-related information from the user. Our approach combines case-based reuse with semi-automated support for retrievals using an intuitive keyword-based method.

III. Interactive Reuse Support

Our approach provides interactive assistance during workflow composition, using CBR based techniques which draw on user-generated annotations to guide retrieval of suggestions. Given a selected annotation framework, authors annotate their workflow inputs and outputs using the framework and add the annotated workflows to the case base, which may already contain workflows annotated by other users.

The framework can assist the user in two different ways:

1) It can retrieve matching workflows, if there are stored workflows with matching inputs and outputs
2) It can retrieve parts of workflows which the user can used to compose his larger workflow

A. Methodology

A workflow contains several services each having one or more inputs and outputs. All the inputs and outputs are annotated with keywords, to provide a description about the input or the output. Within our knowledge-base, we represent these annotated workflow descriptions as cases.

Once the case-base is established, it can be used to identify a workflow or a sub-workflow that matches to the description

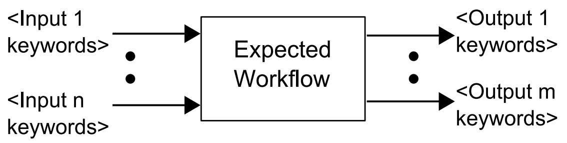


Fig. 1. Input to the system is a set of keyword lists, specifying the inputs and outputs of the expected workflow

and the keywords of the inputs and outputs of the expected workflow provided by the user. (Figure 1).

The system retrieves the possible matches by considering keyword information and the structural information of the workflows stored in the case base. Once the user specifies the search keywords, as shown in Figure 2, the system can match those to a) a service b) a part of a workflow or c) a complete workflow (single output requirement).

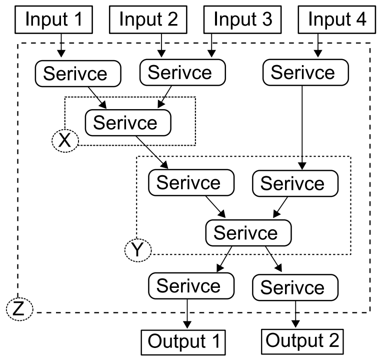


Fig. 2. Matching users' search criteria to parts of workflows

To elaborate on the above, consider the workflow shown in Figure 2. Each box shown by the dashed lines could be a possible match for the user's inputs and outputs. As mentioned earlier, the system takes the structure of the workflow into account when calculating similarity for the user's search criteria. Therefore, for the workflows we consider, which are acyclic, the search is based on the lemma: *All acyclic workflows can be modeled as a graph.* This enables us to reduce the structural search into a graph search.

Depending on the user's requests and the cases stored in the case base, it is not always possible to find a direct match to the user's input and output requirements. Consequently, the system incorporates a simple case adaptation step to fit retrieved cases to new needs. If the system cannot find the required set from a single workflow in our system, it tries to merge the closest workflow with the services in its knowledge base to get the perfect match. For example, user might want the workflow output to be generated as an HTML page, but the most similar workflow in the case base might output an XML file. In this scenario, the system adds to the identified workflow another service that convert XML to HTML fulfilling the user's requirements. The overall operations performed by the system are shown in Figure 3.

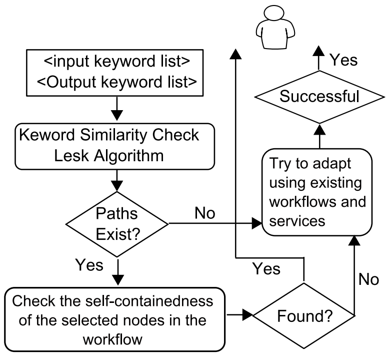


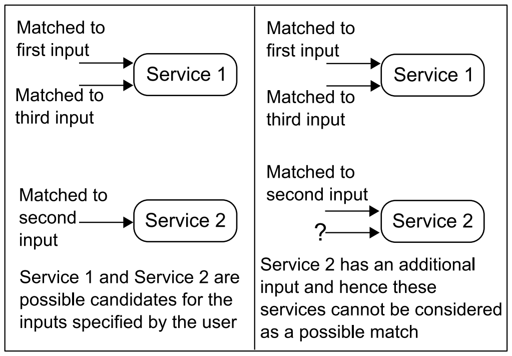
Fig. 3. System and operations and decision points

B. Similarity Assessment

Our framework matches users inputs and output descriptions to the cases, stored in our knowledge base. Before a workflow is stored in the case base, the user is required to annotate the inputs and outputs of each service in the workflow. These annotations can be expressed in simple keywords or semantic languages.

The user specifies the number of inputs and the outputs of the desired portion of the workflow along with a set of keywords for each input and output. The user specified inputs are matched against the inputs of all the services in a given workflow. A match is made when all the inputs of the selected service matches the input descriptions given by the user (Figure 4).

The user specified output is also matched against all the outputs of the services in a given workflow. The inputs and outputs can be matched to different services inside the workflow and hence the system should be able to determine if the selected services are connected. The system also checks the self-containedness of the selected services, using the structural information.



Matched to first input
Matched to third input
Service 1
Matched to second input
Service 2
Service 1 and Service 2 are possible candidates for the inputs specified by the user
Matched to first input
Matched to third input
Service 1
Matched to second input
?
Service 2
Service 2 has an additional input and hence these services cannot be considered as a possible match

Fig. 4. Matching users input descriptions to the inputs of services

Structural Similarity Check : Once a set of input services and an output service are matched for the user's input and output descriptions for a given workflow, the next task that the system performs is to check the structure of the graph. First, it checks if there exists at least one path from all the input services to the desired output service. This will ensure that the selected inputs and outputs are actually a part of a graph and not a disconnected set of nodes. Next it checks whether all the services encountered within these paths form a self contained set of nodes in a workflow (graph) to ensure that the collected set of nodes can be used without any additional inputs to achieve the user's needs. In our system, we store all the workflows as graphs and hence performing the above two checks simply reduces this to a graph search problem.

C. Case Adaptation

In our system, the keyword based similarity check allows the user to specify a certain threshold for the keyword matching. This gives control to relax or tighten the search criteria and hence control the recall. However, sometimes finding exact matches for the user requests only from the stored cases might not be possible. As a solution to this, we incorporated a case adaptation step that is based on the two assumptions:

- the case base stores descriptions of services, which take one input and one output
- the adaptation process considers only these services for constructing extensions to the existing workflows or part of the workflows

The case adaptation uses backward search to adapt a workflow to the user's requests and has the following steps.

- For each workflow, identify input services by running the keyword similarity check using the input descriptions specified by the user.
- Identify services (from the services stored in the case base) which have the output keywords similar to the output description specified by the user.
- For each such service (say service S), find a service in the selected workflow that has the output description similar to the input description of the service S.
- If there is a match, then execute the direct search algorithm to see if the identified input services have connecting paths to the service S and those services in the paths form a self contained set.
- If such a path is found, add the Service S to the end of the identified workflow (or part of the workflow) and present the result to the user.

The above algorithm is not computationally efficient, as it requires searching all the stored cases against the selected workflows. Consequently, a depth limit is used to increase efficiency: The system only performs a few iterations of the above for adaptation.

The final output of a workflow is highly dependent on the specific problem the user trying to solve. For example, a workflow created by a user to retrieve protein structures from an online database may use a service to convert the selected results into an html table for the final presentation. Another user may need to find the exact workflow, but does not need the output in html format, instead needing it in some other format such as an xml file. In such situations, although the case base contains a nearly ideal match, without adaptation the user would not be able to reuse the previous workflow. With our system, if the system can find a service which performs the html to xml conversion, it will be able to adapt the stored workflow to the user requirement and hence suggest it to the user.

IV. Key Benefits

Our approach has a few key benefits. First, such a system saves the user's time, compared to manual techniques alone. Second, compared to automated methods, it requires less processing, because it can rely on the user for focus. Having an ontology might improve the total time and effort, but presents a heavy knowledge acquisition burden avoided by simple keyword-based retrieval. This applies for updates as well: Even if automatic workflow composition tools use pre-indexed information, adding new services can be a challenge.

Our system does not rely on any particular workflow description language [15], [16]. We only consider acyclic workflows in our approach and this enables reducing any workflow description language description into a graph. Our main algorithms work on graph levels and this makes our system independent of any workflow description language.

Our system enables users to provide their input and output descriptions using keywords. For this we used a simple similarity measure based on Lesk Algorithm [17] for keyword matching and disambiguation. Similarity metrics in our system are decoupled from the main system making it easier to plug in a different context matching algorithm. If the semantics are expressed using known standards, again plug-in of similarity components adhering to those standards will be easier as well.

The cases retrieved using our system might not be exact matches to user requirements. But those cases will be intelligent suggestions for the users, who exercise the final judgement. User can set a threshold in the system, to control the quality and the number of results the system provides.

V. Future Directions

Our current system works on graphs and hence supporting a few workflow description languages natively is an important fuure work. Also, in addition to the keyword based similarity model now used for expressing and matching semantics of services and workflows, we would like to support a few semantic languages within the system. We expect to use an existing implementation of those semantic languages and integrate that implementation with our similarity detection and requirement specification components. We also intend to integrate our system into an existing workflow composition framework to help leverage functionality and the usage of those composition tools.

We intend to do a performance evaluation of our system with various knowledge-heavy systems and also with manual systems. Comparing our system performance in terms of memory, speed and response time, as well as in user acceptance, compared to automatic composition frameworks, might provide better insights in to the usability of our system. We note that although we expect our approach to provide certain advantages over both manual and automatic composition systems, it is a complement to them, with each method most appropriate for certain circumstances; it will not completely replace them.

VI. Conclusion

We proposed an approach to supporting workflow composition, leveraging a number of methods from previous research. Our work aims to to improve productivity and the quality of final workflows generated by scientists, by providing alternatives they might otherwise miss and focusing attention on relevant options. We hope that such a framework can help scientists composing workflows to concentrate on their experiments, minimizing the distraction of focusing on the innter details of their workflows.

References

[1] S. Beco, B. Cantalupo, N. Matskanis, and M. Surridge, *Putting Semantics in Grid Workflow Management: The OWL-WS approach.*

[2] D. Martin *et al.*, *OWL-S: Semantic Markup for Web Services*, http://www.w3.org/Submission/OWL-S/.

[3] *The Resource Description Framework*, http://www.w3.org/TR/rdf-syntax-grammar/.

[4] E. Christensen *et al.*, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/wsdl.

[5] D. Leake, "CBR in context: The present and future," in *Case-Based Reasoning. Experiences, Lessons and Future Directions*, D. Leake, Ed. AAAI Press, 1996, pp. 3–30. [Online]. Available: http://www.cs.indiana.edu/ leake/papers/p-96-01.pdf

[6] R. Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. Maher, M. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson, "Retrieval, reuse, revision, and retention in CBR," *Knowledge Engineering Review*, vol. 20, no. 3, 2005.

[7] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic composition of web services using semantic descriptions," in *ICEIS-2003 Workshop on Web Services*.

[8] I. Constantinescu, B. Faltings, and W. Binder, "Large scale, typecompatible service composition," in *IEEE International Conference on Web Services*, 2004.

[9] J. Rao, P. Kungas, and M. Matskin, "Logic-based web service composition: from service description to process model," in *IEEE International Conference on Web Services*, 2004.

[10] M. Carman, L. Serafini, and P. Traverso, "Web service composition as planning," in *ICAPS03 International Conference on Automated Planning and Scheduling*, 2003.

[11] J. L. Ambite and D. Kapoor, "Automatically composing data workflows with relational descriptions and shim services," *International Semantic Web Conference*, 2007.

[12] P. Albert, L. Henocque, and M. Kleiner, "Configuration-based workflow composition," in *IEEE International Conference on Web Services*, 2005.

[13] J. Kim, Y. Gil, and M. Spraragen, "A knowledge-based approach to interactive workflow composition," 2004.

[14] D. Leake and J. Kendall-Morwick, "Towards case-based support for e-science workflow generation by mining provenance information," in *Proceedings of the Nineth European Conference on Case-Based Reasoning*. Springer, 2008, in press.

[15] *Business Process Execution Language for Web Services*, ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf.

[16] *Web Services Flow Language*, http://dps.uibk.ac.at/uploads/100/WSFL.pdf.

[17] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *Computational Linguistics and Intelligent Text Processing*, 2002.