W3C Workshop on Semantic Web for Life Sciences, 27-28 October 2004, Cambridge, Massachusetts USA

Abstracting Workflows: Unifying Bioinformatics Task Conceptualization and Specification Through Semantic Web Services

Nada Hashmi*, Sung Lee†, and Michael P. Cummings‡

*Department of Computer Science, University of Maryland, College Park, MD 20742 USA
†Fujitsu Laboratories of America, College Park, MD 20740-2496 USA
‡Center for Bioinformatics and Computational Biology, University of Maryland, College Park, MD 20742 USA

1 Introduction

Bioinformatics tasks often involve large scale data analysis that requires the integration of results from numerous computational tools, punctuated with data format conversions and human interaction. For high-throughput data analysis, computational tools must be tied together in a coordinated system that can automate the execution of a set of analyses in series or in parallel; that is, a workflow must be created. A simple example, excluding data format conversion and other steps, would be as follows.

> **compose** query sequence
>
> **execute** BLAST to perform pairwise sequence alignment search of appropriate database
>
> **parse** accession numbers of sequences meeting match criterion from BLAST output
>
> **compose** database retrieval requests
>
> **execute** database retrieval requests
>
> **assemble** retrieved sequences
>
> **execute** ClustalW to perform multiple sequence alignment
>
> **process** alignment output to trim non-overlapping sequence regions
>
> **compose** Nexus file
>
> **execute** PAUP* to perform phylogenetic analysis

Thus the methodology for creating workflows relies heavily on the bioinformatics researcher having detailed knowledge and understanding of each tool, and in what order it should be used relative to other tools. This mechanistic orientation of bioinformatics work is in opposition to the logical design of analyses. Conceptually, bioinformatics workflows are conceived in terms of analytical operations necessary to achieve the desired final state. The previous example workflow can be recast to reflect the primary analytical functions, as follows.

> **query** a database of molecular sequences
>
> **retrieve** a set of sequences matching the query
>
> **align** sequences in the set
>
> **generate** a phylogenetic tree based on the alignment

Note that in the example workflow listed above there is no mention of specific tools, data formats and associated format conversions, or other details. Instead the emphasis is appropriately on the analytical operations or functions to be performed rather than the mechanisms for performing those operations. Emphasizing the desired final state, which corresponds more closely to the objective of the research, provides further abstraction and reduces the workflow conceptually to a single functional operation as follows.

> **generate** a phylogenetic tree of sequences homologous to a reference sequence

Here the function "generate a phylogenetic tree" subsumes all the required subsidiary analyses and operations. The specification of a reference sequence, functionally an argument to the operation "generate a phylogenetic tree", is one means of placing bounds on the sequences included (i.e., defining the scope of the operation).

The principal objective of this paper is to suggest a resolution to this disparity between how life scientists

conceptualize their workflows and how the workflows are specified in practice. The resolution is through a process we refer to as *workflow abstraction*: the description of workflows through specification of primary analytical operations or desired analysis final state. Semantic Web technologies will be used to translate from the specification of analytical functions or final state to executable workflows. Thus workflow abstraction results in a unification of workflow conceptualization and specification. We believe such a level of abstraction will be useful to both novice and expert users, and will provide the means to easily and efficiently create workflows to further life sciences research.

The rest of the paper is organized as follows: Section 2 provides background and motivation for our work, Section 3 elaborates our solution for translating workflow abstractions into executing workflows, and Section 4 discusses various issues that must be considered. We conclude in the last section.

2 Background and Motivation

An essential characteristic of the current bioinformatics workflow composition methods is that they are ad hoc: specific workflows are constructed for specific analysis series in a specific manner. This is a necessary consequence of diverse objectives of life sciences research. Typically, a workflow in bioinformatics is performed by a researcher directly executing the required programs to perform the individual analysis steps. Researcher-computer interaction can be reduced in many cases through use of coordinating programs (e.g., Perl or shell scripts) to invoke some of the required analysis programs and perform other necessary operations (e.g., data format conversions, file parsing). However, constructing workflows through use of scripts results in an inherent conflict: the more powerful the script (i.e., the more primary and subsidiary analysis programs it invokes) the less suited it becomes as a component of other workflows. Thus increased power leads to decreased applicability.

Due to the growing need for mechanisms to automate workflows, numerous workflow management systems have emerged that allow users to compose and execute workflows. Workflow management systems seek to provide a partial solution to the power versus applicability conflict, and simultaneously provide a simple and consistent user interface to facilitate workflow construction without the use of scripts or invocation of individual programs. Tools such as Pegasys [8], Taverna [7], Wildfire [11] and BioWBI [5] offer to the user the available tools which are configured into a series and executed at runtime. The user selects individual tools and configures the options for each. Thus current workflow management systems rely heavily on researcher knowledge of the tools and their logical order in a workflow. However, new tools and methodologies are continuously being developed, and researchers may not have the time to become familiarize with all them.

Although workflow conceptualization is function or final state-oriented, all of the workflow systems we have analyzed are resource-oriented. Current workflow systems rely on their own built-in workflow engines which have limited reasoning and automation capabilities, which makes it difficult to specify the functions, automatically compose workflows and ground to available services for execution. These current workflow tools offer semi-automated workflow composition by offering compatible services based on input/output type matches. Although attempts have been made to offer descriptions by manually categorizing the services and sharing related workflows, these methods do not fully overcome the previously mentioned problems, nor do they allow specification of workflows based on functions or desired final state.

3 A Solution

We have sufficiently established the importance of describing workflows in terms of the operations or the desired final state. Although there are many ways of accomplishing the desired functional descriptions we feel the Semantic Web offers the necessary technology to accomplish this task as well as help facilitate the overall goal of automating workflow compositions. The Semantic Web provides meaning appropriate for both machine and human processing. Of the Semantic Web technologies, OWL-S is well-equipped and suited for our purpose. It was created specifically to help in the automatic web service composition and to monitor the execution progress; both of which are very important in the bioinformatics domain. OWL-S has grounding information to the WSDL files for invocation purposes. OWL-S also provides the use of simple processes for providing abstract views of composite processes. This plays a key role in subsuming subsidiary operations described in our example presented in the introduction.

Our previous work experimented with the idea of using OWL-S files to offer better semantic type matches [6]; that is, if tool "A" outputs a protein se-

quence and tool "B" accepts a protein sequence, the two should be made compatible. Data format conversions should be handled by another agent. We focused on an agent to convert the data into a semantic object (a populated ontology) and create a set of rules to handle data exchange among the tools.

We now wish to extend the idea and further abstract the resources which are used to execute workflows. Here we define resources to be the individual programs and the analysis series composed of individual programs. Researchers logically create tasks in terms of functions. These functions are translated into tools being configured in a particular series. This conceptual abstraction insulates the user from the difficulty of data format conversions and tool compatibility. However, this abstraction does not compromise power, and therefore provides both the novice and bioinformatics expert with an appropriate tool with significant advantages compared to the alternatives.

An alternative to using OWL-S would be to create an ontology that would provide accurate and elaborate descriptions of the functions. This ontology would be the layer the user interacts with when composing workflows. Agents would use the ontology to reason and automatically compose of workflows for the user based on simple reasoning heuristics such as use of subclass or disjoint properties. The scenario described by Berners-Lee et al. [1] agent performing a scheduling task is very similar to how ontologies could be used in our case. However, this would be insufficient for the actual translation of the abstract workflow into an executable workflow. Information regarding requirements, execution and transaction completion are all necessary. OWL-S, an OWL ontology to describe services, already meets many of these web service needs.

The task of translating from an abstract workflow to an executable workflow can be accomplished in several ways. One simple method would be through the use of backward composition based on a combination of the "hasInput", "hasOutput", "hasEffect" and "hasPrecondition" properties. Given the function the user wishes to accomplish, the prerequisite of that function will lead to the service that requires prior invocation. The semantic input and output descriptions will further help verify correct compositions. In this way, each service prerequisite will help compose an executable workflow. Subsequently, the workflow will be composed backward and at runtime, reversed for proper execution. Figure 1 shows the method.

It is also possible to move beyond simple "hasInput", "hasOutput", "hasEffect" and "hasPrecondition" combination matching to compose workflows. For example, more sophisticated workflow composition methods would account for information regarding resource availability, user preferences, service ratings and other pertinent information.

4 Prototype Design Choices

Song et al. defined a four-layered architecture to describe a semantic rich environment called Task Computing [9]. This basic architectural design can be modified to describe the workflow abstraction using Semantic Web technology (Figure 2).

User interactions are through the abstraction layer. A GUI-based client can guide the user through entering the desired functionality and multi-layered service specification (*Multi-tiered Workflow Specification*). The GUI-based client is also used to present discovered workflows to users (*Workflow Presentation*).

4.1 Multi-tiered Workflow Specification

Web services in the bioinformatics domain are fundamentally different than typical web services. In typical web services, the outcome is important; e.g. a ticket is purchased, an address is mapped, etc. However, in bioinformatics, the method through which the outcome is reached may be just as important the outcome itself. A researcher may very well be justifiably concerned with what method was employed for the database query, multiple sequence alignment and other similar tasks.

We believe the solution to this problem is to offer multiple layers of user interaction. At the highest level, the user simply specifies a phylogenetic tree is desired with some bounds on taxa/sequences to be included. At an intermediate level the user has the ability to select a specific kind of phylogenetic analysis (e.g., neighbor-joining, maximum likelihood, Bayesian) and parameters (e.g., general time-reversible substitution model, gamma distributed rate variation, invariable sites). At the lowest level, expert users can specify all the analysis parameters for every step in each analysis that constitutes the workflow. These specifications would often include the particular application program that the service represents.

For practical implementations, some reasonable default choices at each step would be offered and users could choose to change any or all of them to suit their

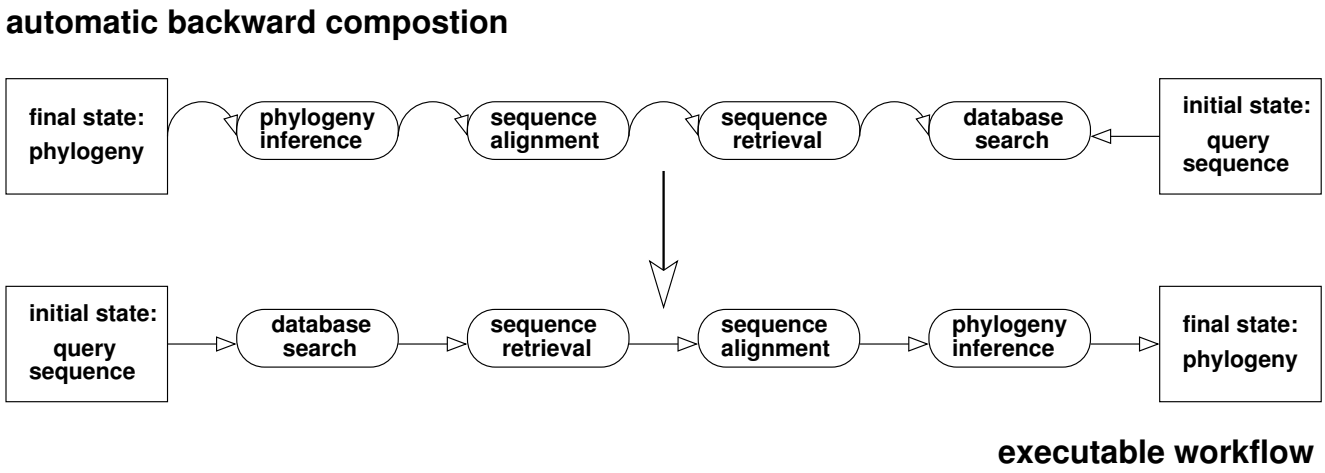

Figure 1: Diagram of workflow composition from user specification of final state through prerequisite matching cascade of service requirements and order reversal to create the executable workflow.

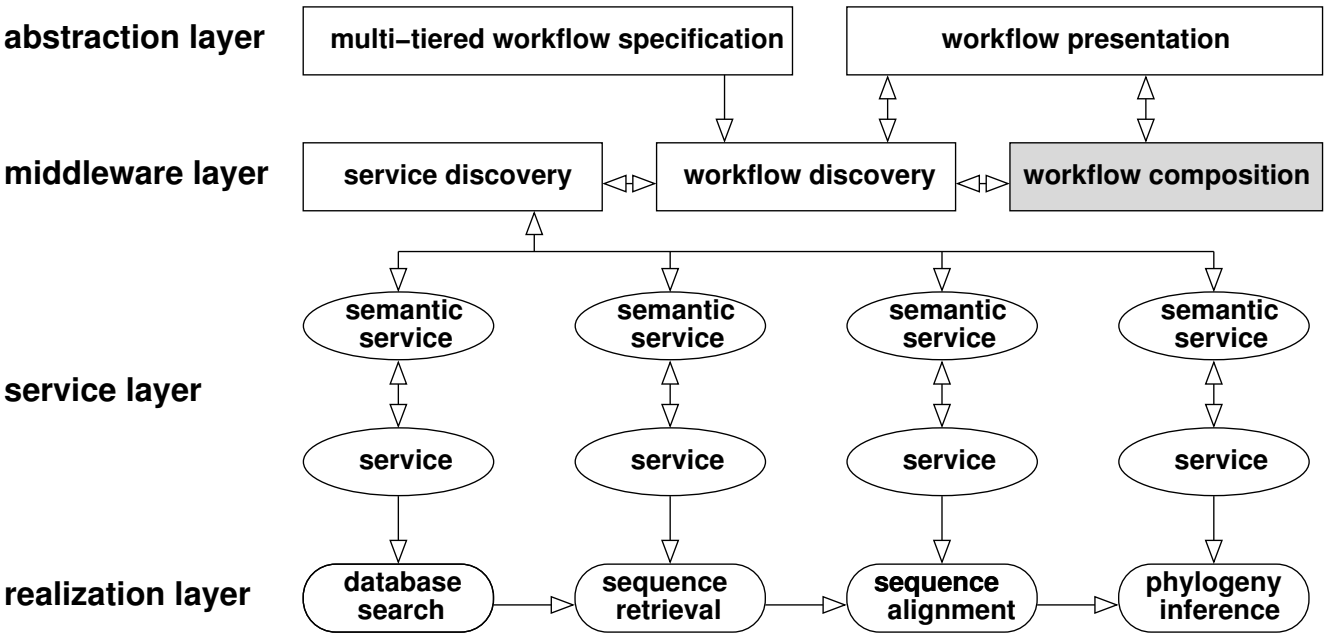

Figure 2: A four-layered architecture for workflow abstraction using Semantic Web technology.

particular needs or wishes. Note that even with full specification there is still a great deal of abstraction, for example the location of the data, the hardware on which the programs will run, etc. This alone is a major step forward compared to the current state of the science. In practice, most users will set their own personal preferences for the individual service choices and associated options, and so in most cases the interaction will return to higher levels of abstraction in subsequent use.

This is the type of solution offered in other domains. Take the example of exercise equipment, say an elliptical machine. Users can get on and just push the quick start button and go without any other interaction, as the machine picks default settings for incline, resistance and time. Users can also pick a particular program (e.g., crosstraining, interval, weight loss). Finally, the user has the option of picking all the parameters if desired.

4.2 Workflow Discovery

Workflow Discovery in the middleware layer is responsible for finding available workflows for the specified functionality. Workflows are discovered using information on their specification, the desired functionality, and available services. For simplicity, we provide an example of workflow discovery using only the desired functionality (e.g., phylogenetic analysis).

The example given in Section 3 produces four potential workflows (Table 1).

If the precondition of service D can be met without invoking service C, service D alone can achieve the final state in workflow 1. Workflow 4 represents the opposite case, where all the preconditions have to be met via invoking other services resulting in a cascade.

We must define the term *available workflow*. One approach is to consider a workflow available regardless of the availability of constituting services. For example, workflow 4 would be advertised as available even when service A is not available. In this case, it is up to the user to satisfy the precondition of service B. This approach makes the workflow discovery logic simple. However, the burden is now on the user to supply the missing component.

Another possibility is to consider a workflow available only if all constituting services are available. Thus, when service A is not available, only workflow 1–3 are advertised as available. The downside of this approach is that discovering and maintaining workflows could be expensive and may not scale well.

Additionally, we have the option of showing the details of the workflow to the user.

4.3 Workflow Composition

The execution of a workflow depends on the service composition: explicit or implicit. In explicit composition, the services that make up the workflow are composed and published as an encompassing service, responsible for triggering their execution in the right sequence. In implicit composition, a utility service such as *satisfyPrecondition* is called at the beginning of each process model for services with preconditions. As a result, the prerequisite services are executed first, in order to satisfy the preconditions; once all preconditions are met, the initial service is finally executed. The precondition dependency check is recursively applied until a service with no preconditions is reached.

4.4 Semantic Service Description

Services corresponding to tools and devices in the realization layer can be defined as Web Services with WSDL, and semantic services can be defined with OWL-S. A semantic service description is constructed with a precondition that reflects the prerequisites needed in order for the service to successfully complete; this precondition is sufficient to construct workflows for the desired functionality.

We have considered two ways of defining service preconditions. For example, a phylogenetic analysis service, which requires a multiple sequence alignment as an input, can be first described as follows.

Service: Phylogenetic Analysis (PA)

> **Input:** Optional multiple sequence alignment and optional analysis parameters
>
> **Output:** Phylogenetic Tree
>
> **Precondition:** Multiple sequence alignment, if not given as input

Alternatively, the same service can be viewed as the union of the following two separate services.

Service: Phylogenetic Analysis (PA-1)

> **Input:** Multiple sequence alignment and optional analysis parameters

Table 1: Four potential workflows for example given in Section 3

Workflow 1:	Service D			
Workflow 2:	Service C	+ Service D		
Workflow 3:	Service B	+ Service C	+ Service D	
Workflow 4:	Service A	+ Service B	+ Service C	+ Service D

Output: Phylogenetic Tree

Precondition: None

and

Service: Phylogenetic Analysis (PA-2)

Input: Optional analysis parameters

Output: Phylogenetic Tree

Precondition: Multiple sequence alignment

The benefit of the first approach (PA) is that it is concise. However, we believe that it may introduce additional burden on workflow discovery since the workflow discovery engine may have to store more state information for all branches. The second approach (PA-1 $\cup$ PA-2) is more efficient in terms of workflow discovery. However, this approach may lead to an increased number of small semantic services, thus making publishing and maintaining services more difficult for providers. This issue can be resolved through a utility tool that can automatically generate and publish PA-1 and PA-2, given PA.

Bioinformatics computational tools such as Phylogenetic Analysis, Multiple Sequence Analysis, and Database Search carry out tasks at the realization layer. Physical devices (e.g., microarray readers, DNA sequencers) that produce data are also potential residents of this layer, typically through associated computers that provide points of control and data collection.

4.5 Challenges

Workflows are discovered primarily using properties defined in semantic service descriptions, such as preconditions, service category and rating, etc. One difficulty would be making sure that definitions can be understood across independently developed services. We believe that adopting a well defined standard such as Life Sciences Identifier (LSID) [4] and popular ontologies such as GeneOntology [2], BioPAX [3], and National Library of Medicine's UMLS [10] along with the use of ontology translators can help us address this issue. Natural Language Processing can also be used to bridge service descriptions.

We presented at least two alternative design choices for different components. Each design choice has implications — usability versus performance (thus leading to scalability issues). The challenge is to find a solution that offers a good compromise.

As with any other service oriented architecture, we have to trust the functionality and capability advertised by each service. However, in order to mitigate the risks inherent in using external services, we will consider building a trust model into the workflow abstraction.

5 Conclusions

In current practice there is a disparity between bioinformatics workflow conceptualization and specification. We believe this disparity can be resolved through a process we refer to as workflow abstraction: the description of workflows through specification of primary analytical operations or desired analysis final state. Semantic Web technologies can be used to translate from the specification of analytical functions or final state to executable workflows. Thus workflow abstraction results in a unification of workflow conceptualization and specification. We believe such a level of abstraction will be useful to both novice and expert users, and will provide the means to easily and efficiently create workflows to further life sciences research.

Acknowledgments

We thank Adam Bazinet, Ryusuke Masuoka, and Stephen McLellan for helpful comments and suggestions on the manuscript.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 279:34–43, 2001.

[2] Gene Ontology Consortium. http://www.geneontology.org/.

[3] Biological Pathways Exchange. http://www.biopax.org/.

[4] Life Sciences Identifier. http://www-124.ibm.com/developerworks/oss/lsid/.

[5] Bioinformatic Workflow Builder Interface. http://www.alphaworks.ibm.com/tech/biowbi.

[6] R. Masuoka, B. Parsia, and Y. Labrou. Task Computing — the Semantic Web meets pervasive computing. *Lecture Notes in Computer Science*, 2870:866–881, 2003.

[7] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, Jun 16, 2004.

[8] S. P. Shah, D. Y. M. He, J. N. Sawkins, J. C. Druce, G. Quon, D. Lett, G. X. Y. Zheng, T. Xu, and B. F. F. Quellette. Pegasys: software for executing and integrating analyses of biological sequences. *BMC Bioinformatics*, 5:40, 2004.

[9] Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in Task Computing. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, pages 310–318, Boston, Massachussets, USA, 2003.

[10] Unified Medical Language System. http://www.nlm.nih.gov/research/umls/.

[11] Wildfire/GEL. http://web.bii.a-star.edu.sg/∼francis/wildfiregel/.