

Improving the Reuse of Scientific Workflows and Their By-products

Xiaorong Xiang Gregory Madey

Department of Computer Science and Engineering

University of Notre Dame

xxiang1, gmadey@nd.edu

Abstract

Most current practical methodologies and workflow systems for service composition and workflow creation in e-Science pursue a semi-automatic way to allow users to discover and select appropriate services to include in a workflow based on semantic and conceptual service definitions. However, few of these approaches consider the potential for reuse: to share the knowledge gained during the service composition process and to reuse complete or partial of existing workflows. We believe that providing a capability for reuse of this knowledge and workflows could be an important component in a workflow system. In this paper¹, we present a methodology and an enhanced system design to facilitate the reuse of knowledge and workflows. It contains a hierarchical workflow structure representation, knowledge management and knowledge discovery components to capture and manage the reusable information in a system, and an approach for using a graph matching algorithm to discover similar workflows.

1 Introduction

As more data, analysis tools, and other resources are delivered as services on the web, the major benefit of adopting service-oriented architecture in e-Science, is that of allowing scientists to describe and enact their experimental processes by orchestrating distributed and local services into a workflow. It often involves choosing a set of appropriate services based on the functional and non-functional properties of services, ordering them in sequence, resolving connectivity between the services, and converting the complex process into a target workflow language that can be deployed and invoked on a platform. Over the past several years, much research has been done on approaches for service discovery and composition in order to achieve the goal of seamless web service composition [13]. A significant

portion of the work aims at automating discovery and composition by combining ontological annotation of services and AI planning technology. In the literature, the demonstration of these approaches is largely applied on virtual travel agencies or small well-defined domains. Applying these approaches to larger, more complex and less-defined applications can be difficult, especially before a complete strong ontological agreement is established in the application domain or across multiple domains.

Most current practical methodologies for service composition or workflow creation employ a semi-automatic design that allows users to discover and select appropriate services to include in a workflow based on semantic and conceptual service definitions. This partially lifts the load on the users of requiring detailed knowledge and understanding of each tool, service, and data type. In the meantime, it increases the complexity of building such middleware to support workflow creation at a higher level abstraction. Several workflow management systems and service-oriented middleware, such as Pegasus [5], myGrid/Taverna [14], Kepler [11], and Triana [19], are developed and with the intent to streamline the workflow design, execution, monitoring, and re-run the workflow.

Most of these systems and approaches provide users an environment to compose services from scratch in terms of more accurately choosing appropriate services with consideration of semantic matching and quality of services (QoS). Fewer of them consider the potential of reuse and sharing of the knowledge gained during the service composition process and reuse of complete or partial existing workflows. We believe that providing a capability to reuse the knowledge and workflows is an important component in such a system. This reusability will lead to a more efficient and more structured composition process that will accelerate rapid application development. It will provide more valuable guidelines to assist users with their workflow creation using knowledge that has been gained and verified by others. Reuse of the verified knowledge will potentially increase the correctness of composed workflows and reduce the errors that may be caused by misannotation, inaccurate

¹Supported in part by the Indiana Center for Insect Genomics, with funds from the Indiana 21st Century Research & Technology Fund

annotation, and incomplete annotation of services. The requirement of complete information about the world brings challenges of applying traditional AI planning technologies into the service composition process since it is not feasible nor possible to collect all the information to form a complete initial state of the world [8, 13, 10]. The gradually gathered knowledge in the system during service composition process may help accumulate more complete information for an AI planner.

In this paper, we present a methodology and an enhanced system design to facilitate the reuse of knowledge and workflows. It contains a hierarchical workflow structure, knowledge management and knowledge discovery components to capture and manage the reusable information in a workflow system, and an approach for using a graph matching algorithm to discover similar workflows. The methodology proposed is being used in the design and implementation of a service-oriented based system for supporting bioinformatics research.

2 A hierarchical workflow structure

We define a hierarchical workflow structure that contains four levels of representation (see Figure 1): abstract workflow, concrete workflow, optimal workflow, and workflow instance.

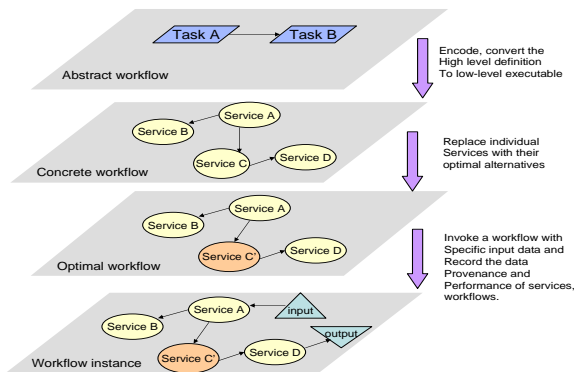


Figure 1. A four level hierarchical workflow structure representation and transformation of scientific processes

Abstract workflow is a definition of a scientific process with emphasis on the analytical operations or function to be performed rather than the mechanisms for performing these operations. An abstract workflow may be defined by an end-user using standardized syntax,

vocabularies, and semantics developed in their scientific communities. Users logically create each task in terms of functions they wish the task should accomplish.

Concrete workflow is a definition of a number of tasks represented as actual executable services. A concrete workflow can be converted to specific workflow language and sent to a workflow engine to be executed. The **translation** of an abstract workflow into a concrete workflow is a process of discovering suitable services that implement these functions and solving the connectivity between services.

Optimal workflow is a concrete workflow where individual executable services are replaced by alternatives with highest quality. The **optimization** of a concrete workflow into an optimal workflow is a process of ranking services based on a set of metrics and selecting an optimal service to replace each service in the workflow.

Workflow instance is an actual run of a concrete workflow or optimal workflow with input data and generated output data. Since a scientific process is a process for discovering new knowledge, keeping track of the source of a workflow result can be as important as the result itself. Many data provenance systems in e-Science [18] have focused on recording the data from which a data product evolved and the process of transformation of these data, i.e., input data, output data, and process. We believe that recording the information of running time and failure rates of each workflow instance can provide measurements for profiling the quality of services and workflow. This information can be used to assist the workflow optimization process.

Several benefits are provided with this hierarchical workflow structure definition:

Allows users to define workflow at different abstract levels. Less experienced users may define a workflow in terms of functions they wish a task should perform. Intermediate users may define a workflow with more detailed properties of each task, such as the algorithm and data source they may want to use. Expert users may be able to define a workflow in an ad-hoc approach by choosing appropriate executable services and form a workflow with appropriate logic. Users would like to conduct an experiment to determine “*if gene genealogies for ATP subunit α, β, γ are different*”. We assume that there are four services in a system and they are annotated using predefined vocabularies in the bioinformatics domain. Users with different knowledge level of the system and the particular domain may define the workflow differently as shown in Figure 2.

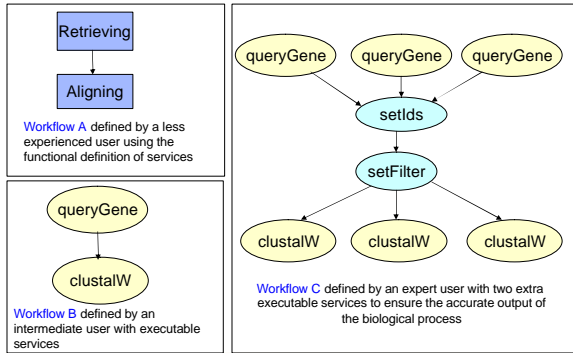


Figure 2. An example illustrates the user-oriented workflow definition with different levels of knowledge

Allows the transformation of workflows in semi-automatic or automatic ways. The transformation from abstract workflow to the concrete workflow can be completed by an expert bioinformatician with assistance from a service discovery agent provided in a system, such as the myGrid/Taverna [14], IRIS [16], and the BioMoby [7] projects.

The transformation from the concrete workflow to the executable workflow can be completed automatically by ranking services and choosing optimal ones. Most of previous work bound the information regarding the quality of service with the translation process results in more sophisticated and complex composition methods. Since most measurements of the quality of services are dynamically changing, this tightly-coupled representation and composition method is not easily adapted to these changes. Separating the optimal workflow from the concrete workflow allows the easy integration of Grid computing technology to address the resource allocation and security issues of data and computation resources.

Allows the full or partial reuse of workflows defined at different levels. Reuse of a workflow may occur when users need to replicate their data sets or rerun the same workflow using different input data. For example, consider a scientist who is interested in a data set generated from a given workflow. Using the recorded data provenance, the corresponding concrete workflow that was used to generate this data set can be discovered. The concrete workflow can be re-optimized and invoked with different input data.

The reuse of a workflow may also occur during workflow design. For example, a scientist may have a high level representation or partial representation of a workflow, searching the workflow repository may return a number of similar workflows at an abstract level and/or concrete level. This scientist may choose a candidate to reuse or modify the

workflow to meet the goal.

3 An enhanced workflow system

A general workflow system contains following components to support the semi-automated workflow composition or automated composition process.

- Ontologies serve as a common vocabulary for semantic annotation of services and data in the system. As most current semantic web services standards are relatively mature and stable, the ontology model used in a system is built upon a distributed and modularized ontology structure and reuse some cross-domain ontologies. The use of a well-defined ontology could potentially increase the interoperability for information published on the web. The ontology model used in a system normally contains two modules: *generic service description ontology*, such as OWL-S, is an ontology module used to specify generic web service concepts; *service domain ontology* is an ontology module designed and used for the semantic description of web services in a particular domain.
- Semantics enabled service registry is responsible for storing the semantic and syntactic information of services as well as answering the inquiry. The semantic information can be provided by service providers or third party annotation.
- Workflow composer (human expert and/or software agent) discovers appropriate services and resolving the connectivity between services. It is also responsible for converting the workflow into a workflow language that can be executed on a workflow engine.
- Data provenance management keeps track of the origin of the data products.

Few workflow systems have the capability for the reuse of the knowledge gained during the service discovery, service composition, and service invocation process. We add two components – knowledge discovery and knowledge management – to the workflow system illustrated in the Figure 3 and discuss how this knowledge can be used over time to provide more accurate guidelines to users.

We give a definition of service in our system as a tuple with several important attributes:

$service_i(description_i, operation_i, \dots)$ – a service contains text descriptions of its feature, a set of operations (must not be \emptyset), and other attributes;

$operation_{ij}(description_{ij}, input_{ij}, output_{ij}, quality_{ij}, performtask_{ij}, \dots)$ – an operation in a service contains text descriptions of its features, a set of input parameters (may be \emptyset), a set of output parameters (may be \emptyset), a set of quality metrics, semantic

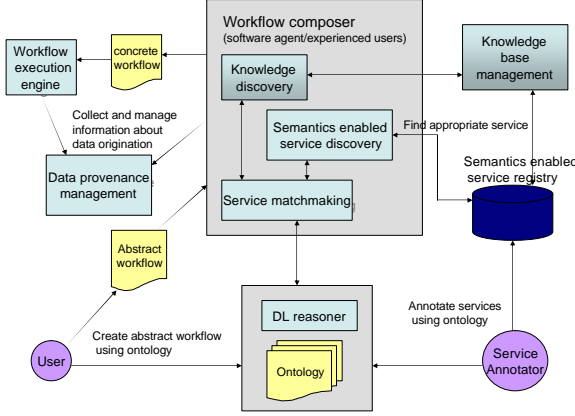


Figure 3. An enhanced workflow system with two added components, knowledge management and knowledge discovery

description of the features using vocabulary from *service domain ontology*, and others;

$parameter_k(semantic_k, datatype_k)$ – a parameter contains semantic description using vocabulary from *service domain ontology* and the data type.

Since our system is intent to be used in bioinformatic application, the adopted *service domain ontology* is from the myGrid project (<http://www.mygrid.org.uk/ontology>) with extension of a self-define module (the *application ontology*) that contains vocabularies particularly used in our system. The semantic annotation of services and workflows can be represented as a RDF model and stored in a RDF repository.

3.1 Knowledge management

The knowledge management component is responsible for collecting, analyzing, and handling inquiries on the knowledge base. The knowledge base holds information gathered incrementally during workflow translation and service composition processes. This information provides increasingly accurate guidelines for users over time. Four types of information are classified:

- **Connectivity of services.** A concrete workflow can be viewed as a graph with a number of linked services in a certain order and logic. Each node in the workflow is an operation of an executable service. In a simple case, two nodes are connected if an output parameter of one operation maps an input parameter of another operation based on their syntactic and semantic description.

Rule 1: $operation_{ij} \rightarrow operation_{mn}$ if $\exists parameter_k \in output_{ij}$ and $\exists parameter_o \in input_{mn}$ and $datatype(parameter_o) = datatype(parameter_k)$ and $semantics(parameter_o) = semantics(parameter_k)$

Rule 2: if $operation_{ij} \rightarrow operation_{mn}$ then $service_i \rightarrow service_m$

While the connectivity of services can be determined by these above simple rules, it can be identified using more complex models [12].

The connectivity between two services can be identified automatically based on the rule defined above when a new service is added to the system. It is a computationally intensive process when the number of services in the system and the number of parameters for each operation is large. Also, incorrectly identifying the connectivity between two services is most likely introduced by the misannotation of services or an incomplete ontological model. Therefore, during the translation process, the connectivity structure should be refined and updated based on human judgment. After a concrete workflow is created and verified, the connectivity of services in the workflow can be added into the system. As time goes by, the connectivity of services in a system forms a graph of the knowledge space. A vertex in the graph can be represented as $(service_i, operation_{ij}, parameter_{ijk})$ or $(service_i, operation_{ij})$ if one operation does not have parameters and the edge represents the connectivity of two vertices.

- **Alternativity of services.** In the context of our research, we define $service_i$ as an alternative of $service_m$ if $\forall operation_{ij} \in service_i$ and $operation_{mn} \in service_m$ their syntactic and semantic description are the same except the quality properties. For example, two services that implement the same WSDL interface are alternatives for each other. These two services may implement the WSDL interface using different underlying technologies, charging different fees, and having different performance.

The execution of workflows and services takes place in a distributed computing environment. The execution may fail at some point due to the failure of the workflow engine, failure of the service, and failure of the network fabric. The capability to dynamically select alternative services ensures the recovery from service failure. The myGrid/Taverna project provides users a way to encapsulate alternative services into the workflow at the design time. Another approach is to find an alternative service during run time using general semantic service discovery technologies. We believe that identifying and storing the alternatives of a service ahead of time can increase the performance by eliminating this semantic service discovery process. The method can also improve the correctness of finding alternative services. The alternativity of services can be automatically identified when a new service is added in

the system and be refined during the workflow translation process.

- **Quality profile of services.** As more services with similar functionalities are published, it is important to define qualitative metrics that help the selection of the optimal services. Modeling the quality of service and approaches for choosing optimal services has been well studied for several years [1]. While there are a number of quality criteria that can be used for ranking services, different systems choose different sets of metrics and quality models for computing the overall quality of service. We define quality with four attributes.

Quality(cost, trustness, executiontime, failure rate)
– *cost* is the fee needed to execute an *operation_{ij}* and it is provided by the service provider;
– *trustness* defines users preference of using the *operation_{ij}* based on their experiences and it is annotated by users;
– *executiontime* and *failure rate* define the performance of an *operation_{ij}* and they are collected and calculated from each run of a workflow or service.

Other QoS properties, such as security, may also be added when needed. The overall quality of each service can be computed periodically or during the optimization process using the similar QoS computation model algorithm defined in [9].

- **Mapping between abstract workflow and concrete workflow.** The construction of the abstract workflow represents the knowledge that scientists knows about their domain and the services/tools provided in the system. The abstract workflow and the semantic annotation of the concrete workflow are represented using the ontology. The concrete workflow also is represented using a particular workflow specification that can be invoked on the workflow engine. Recording the mapping relationship between abstract workflow to concrete workflow enables finding similar workflows in the system given a workflow in different representation format. The concrete workflow can have its own semantic annotation. It can also be represented using the specific workflow language that can be invoked in the workflow engine.

3.2 Knowledge discovery

The knowledge about the connectivity of services, alternativity of services, quality of services, and workflow representations is typically stored in tables. The knowledge discovery component resides in the workflow composer. It is responsible for communicating between the workflow composer and the knowledge management component during

the workflow translation process to find appropriate knowledge in the system. It is also responsible for selecting and replacing services with their optimal alternatives during the optimization process and to find a replacement during run time. The knowledge discovery component accepts and sends requests to the knowledge management component.

4 Translation process

The process of translating abstract workflow into a concrete workflow involves the discovery of appropriate services and resolving the connectivity between services in order to accomplish tasks defined in the abstract workflow.

4.1 Service discovery and matchmaking process

During the translation process, the workflow composer issues a query to find appropriate services that can be used to accomplish the defined task. For example, the composer is interested in finding an operation which performs the task “aligning”. A general query returns all services having #performTask property equals #aligning. More sophisticated discovery processes use reasoning capabilities to infer a subsumption relationship between the requested service and the services described using the ontology.

The translation from an abstract workflow to a concrete workflow requires solving the connectivity between two executable services with mismatched or inappropriate input to output. The mismatching problem may be introduced by inaccurate semantic annotations, incomplete semantic annotations, and inaccurate ontological reasoning. The false positive (FP) is a case that the connectivity of two services is automatically identified as matched but these two services can not be connected together in reality. This type of error can be detected by experts at design time or be identified after the formed workflow runs and incorrect results returns. The true negative (TN) is a case that the connectivity of two services is identified as mismatch and in reality they are not matched. Adaptor, shim, or mediator [16] technologies are used to align or modify poorly typed input and output of consecutive services in a workflow. These mediators are stored in mediator pools and discovering such a mediator is achieved with ontologies and machine reasoning, the same as the discovery of normal services.

4.2 Knowledge reuse

With the incrementally added information in the knowledge base, solving connectivity can be done completely at the syntax level without need for consulting the domain ontology. As time goes by, converting the abstract workflow

to the concrete workflow may be achieved by finding a mediator between two services in the knowledge base. Thus, the use of ontologies will be exactly on those parts of the workflow that were never used before. The manual translation process will be required just once for every new element of the set of components in a workflow and when a new service is added in the registry. The problem of solving the connectivity between two services can be converted to a problem of finding a path between two nodes in a connectivity graph. During the translation process, instead of resolving the connectivity from scratch using semantic reasoning technology, the composer can reuse stored knowledge to support the semi-automatic and automatic composition.

1. Given a service or operation, all services or operations connected to the current service or operation can be found by table lookup and presented to the users. Users can choose one based on their expertise. Since the connectivity stored in the table is verified during the previous workflow creation process, we expect the probability of finding an accurate one is higher and faster than using the semantic reasoning techniques from scratch.
2. Given two services or operations, find one or a sequence of services or operations between them (mediators) that can connect these two services or operations together. This problem can be converted to a problem of finding a path between the service or operation A to the service or operation B. Since the connectivity structure of services or operations in the knowledge base is a graph, the shortest path algorithm (Dijkstra) is applicable to this problem.
3. This approach can be extended into a wider use case when users know the exact input they can provide and output they are trying to get. A general planning technology is trying to find a service or operation that accepts this input and a service or operation that generates this output. Using the connectivity structure, the path between the input and output can be found, if there is any.

4.3 Implementation and evaluation

The connectivity between two services is identified automatically when a new service is registered into the semantic-enabled registry using the matching rules defined in the Section 3. As more services are registered in the registry, the larger connectivity graph is formed. Since the automatic identification process may introduce some mismatching problems, the mismatching cases can be corrected during the workflow translation process with knowledge from experts.

Table 1. PERFORMANCE EVALUATION OF MATCH DETECTION PROCESS

Number of Services	Number of Matched Pairs	Load RDF repository (milliseconds)	Average time of match per single service (milliseconds)
200	10	1547	12.02
400	34	2346	13.01
600	84	2600	12.31
800	138	3015	12.35
1000	225	3325	12.51

The connectivity graph approach is evaluated on an Dell laptop with a 1.5GHz Pentium M CPU and 512M of RAM. Service descriptions are randomly generated using 418 concepts from domain ontology for semantic type and 10 concepts for data type. Each service contains 1 operation. Each operation has 1 input and 1 output. Additionally, we annotate 4 services that can be used to compose a workflow to conduct the experience defined in 2. These services are added into the randomly generated data set in order to verify the correctness of our approach.

The measured performance of the match detection process during the service registration process is reported in Table 1. The number of matched pairs reports the identified pair of services that one service's output can be fed as input for the other service. Although the time of loading the RDF repository (implemented with the Sesame 1.2.6 (<http://www.openrdf.org/>)) increases as the number of generated services increases, the process is typically only need to be done once. The average time of the matching process when a new service is registered in the repository requires about 12-13 milliseconds.

The searching function of shortest path algorithm is evaluated using the connectivity graph created from 1000 randomly generated semantic web services. The graph is constructed with matched pair and input/output of each service in matched pair. The load time of the connectivity graph (724 nodes and 587 arcs) is about 220 milliseconds and only need to be done once the system is loaded. The average path searching time is less than 1 milliseconds. Using this search function, the 4 real world services forms a workflow as we expected.

The preliminary results show that the feasibility of our implementation is acceptable. Further testing with real services/workflows and larger number of input/output parameter and operations is needed .

5 Workflow reuse

Both abstract workflow, and concrete workflow can be viewed as a graph. With this type of graph representation,

graph matching techniques can be applied to find similar workflows in the system. Although in-depth graph theoretic research is not the main focus of this investigation, we are interested in applying an efficient algorithm to find similar workflows in the system given the graph representation of abstract workflow or concrete workflow.

SUBDUE (available at <http://cygnus.uta.edu/subdue/>) is a graph-based knowledge discovery system that finds structural and relational patterns in data representing entities and relationships. SUBDUE represents data using a labeled, directed graph in which entities are represented by labeled vertices or subgraphs, and relationships are represented by labeled edges between the entities. The SUBDUE graph match utility [2] is a part of the SUBDUE data mining system. The graph match utility can perform exact and inexact graph matches on directed or undirected graphs with labeled vertices and edges.

For example, a scientist may have a scientific process in her mind such as: *“I’d like to get all ATP alpha units of plastids in my MoG investigation and do multiple sequence alignments and get an alignment report with a format that I am able to feed into my local PAUP program.”* A possible abstract workflow she may define is similar to Figure 4.

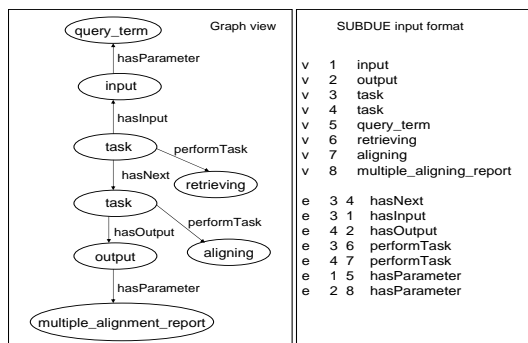


Figure 4. The graph representation of a workflow for describing a scientific process

The given workflow is converted to the graph representation that can be fed into the match algorithm. The match algorithm computes the similarity of the given workflow against all the workflows stored in the knowledge base. The returned match cost from the SUBDUE algorithm is the measurement that we use to rank the similarity of the workflows. If two graphs are identical, the match cost is 0. Costs of various graph match transformation have effects on the results. The costs can be changed based on the importance of each transformation. For example, we might like to define that the cost of substituting a vertex label or edge label is higher than the cost of deleting the vertex or edge. With

this specified, the algorithm can find more optimal results.

The threshold of returned workflows is defined based on the match cost. One or more workflows with the most similarity are returned and presented to users. Users may decide to use these workflows as a template to manipulate their workflow definition on the abstract level or concrete workflow level. Alternatively, users may decide to use the returned workflow to conduct their experiments.

6 Related work

Abstract and concrete workflows have been introduced in various scientific workflow literature and systems [3, 4, 15]. These two representations create a view of certain aspects of a workflow that meet the interests of users with different knowledge levels of the services and a particular domain. However, in these systems and literature, the notion of concrete workflow and optimal workflow are combined together and are often not distinguished as two separate representations. We believe that separating these two workflow representations provides flexibility of dynamic binding, the ability to select optimal services, and easier integration of Grid resource management services.

The translation of an abstract workflow into a concrete workflow is a process of service discovery and service composition. It normally uses an ontology to annotate services and applies reasoning and matchmaking technologies from a workflow. A number of research investigations focus on automation of this process and assume that the ontological model is well defined and services are correctly annotated, which is not always the case. Rao et. al. [17] presents an approach that addresses the reality of incomplete annotation. The framework helps users become better at annotating composable functionality over time. The enhanced system and methodology proposed in this paper is intended to reuse the knowledge that has been verified by others. It provides users more accurate guidance for service discovery using that stored knowledge.

The importance of reuse and repurposing of workflows has been reported in [20]. Antoon Goderis et. al. [6] presents an approach of using graph based solution to find similar concrete workflow on the web. This is a similar approach similar to what we used but with a different graph matching algorithm and different graph representations.

7 Conclusion and future Work

In this paper, we present the importance of implementing workflow and knowledge reuse. In order to support that reuse, we propose and describe a methodology and an enhanced workflow system. It includes a hierarchical workflow structure consisting of four levels that allow users to

specify workflows at different levels of abstraction, based on their knowledge and experience. Two components are added into the workflow system to collect and analyze the reusable of information generated from the service registration and the workflow translation process.

The methodology proposed is being used in the design and implementation of a service-oriented based system for supporting bioinformatics research. It can also be used in workflow systems in other domains. Based on its successful design and implementations of the system (MoGServ) [21], we developed an ontological model for data and services annotation in the system. At the current stage, the number of services, operations, workflows in the system is relatively small, but are expected to grow with usage. The future MoGServ is intended to support genomic research and provide a workbench for biologists in the Indiana Center for Insect Genomics (ICIG)², a research center composed of three academic institutional partners. Users can define a genomic research workflow through a web interface for a particular application. It may result in higher productivity for genomics researchers and synergy resulting from transparent integration of data and analysis tools from multiple locations. We believe that the enhanced workflow system with the knowledge reuse capability can provide more accurate guidelines during the workflow creation process and make the process more efficient as time goes by. A systemic evaluation is being conducted.

References

- [1] P. A. Bonatti and P. Festa. On optimal service selection. In *Proceedings of the 14th international conference on World Wide Web*, 2005.
- [2] D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.
- [3] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbre, R. Cavanaugh, and S. Koranda. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1), 2003.
- [4] L. A. Digiampietri, C. B. Medeiros, and J. C. Setubal. A framework based on web service orchestration for bioinformatics workflow management. *Genetics and Molecular Research*, 4(3):535–542, 2005.
- [5] Y. Gil, E. Deelman, J. Blythe, C. Kesselman, and H. Tangmunarunkit. Artificial intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems, special issue on E-Science*, Jan/Feb 2004.
- [6] A. Goderis, P. Li, and C. Goble. Workflow discovery: the problem, a case study from e-science and a graph-based solution. In *IEEE International Conference on Web Services (ICWS'06)*, 2006.
- [7] E. Kawas, M. Senger, and M. D. Wilkinson. Biomoby extensions to the taverna workflow management and enactment software. *BMC Bioinformatics*, Nov. 2006.
- [8] U. Kuter, E. Sirin, D. Nau, B. Parsia, and J. Hendler. Information gathering during planning for web service composition. In *The Third International Semantic Web Conference (ISWC2004)*, Hiroshima, Japan, 2004.
- [9] Y. Liu, A. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In *WWW2004*, 2004.
- [10] S. Lu, A. Bernstein, and P. Lewis. Automatic workflow verification and generation. *Theoretical Computer Science*, 353(1), 2006.
- [11] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039 – 1065, Dec 2005.
- [12] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 2003.
- [13] N. Milanovic and M. Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51–59, November/December 2004.
- [14] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17), 2004.
- [15] U. Radetzki and A. B. Cremers. Iris: A framework for mediator-based composition of service-oriented software. In *2004 IEEE International Conference on Web Services (ICWS 2004)*, July 2004.
- [16] U. Radetzki, U. Leser, S. Schulze-Rauschenbach, J. Zimmermann, J. Lussem, T. Bode, and A. Cremers. Adapters, shims, and glue—service interoperability for in silico experiments. *Bioinformatics*, 22(9):1137–1143, 2006.
- [17] J. Rao, D. Dimitrov, P. Hofmann, and N. Sadeh. A mixed initiative approach to semantic web service discovery and composition: Sap's guided procedures framework. In *Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 401 – 410, 2006.
- [18] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(5), Sept 2005.
- [19] I. Taylor, M. Shields, I. Wang, and A. Harrison. Visual Grid Workflow in Triana. *Journal of Grid Computing*, 3(3-4):153–169, September 2005.
- [20] C. Wroe, C. Goble, A. Goderis, P. Lord, S. Miles, J. Papay, P. Alper, and L. Moreau. Recycling workflows and services through discovery and reuse. *Concurrency and Computation: Practice and Experience*, 2007.
- [21] X. Xiang, G. Madey, and J. Romero-Severson. A service-oriented data integration and analysis environment for in-silico experiments and bioinformatics research. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, January 2007.

²<http://ctdrt.bio.nd.edu/index.php?content=projectinfo.php&projectno=4>