

A Service-oriented Data Integration and Analysis Environment for *In Silico* Experiments and Bioinformatics Research

Xiaorong Xiang Gregory Madey
 Department of Computer Science and Engineering
 Jeanne Romero-Severson
 Department of Biological Science
 University of Notre Dame
 xxiang1, gmadey, jromeros@nd.edu

Abstract

In this paper, we present a practical experiment of building a service-oriented system upon current web services technologies and bioinformatics middleware. The system allows scientists to extract data from heterogeneous data sources and generate phylogenetic comparisons automatically. This can be difficult to accomplish using manual search tools since sequence data is rapidly accumulating. A web-based environment enables scientists to more effectively define a task, perform the task at a desired time, monitor the execution status, and view the results. The first prototype of this system is being evaluated on a phylogenetic research application, Mother of Green (MoG). Our evaluation demonstrates that a service-oriented architecture can accelerate scientific research, increase research productivity, and provide a new approach to doing science. We also discuss issues in design and implementation of the system and identify our future research directions to enhance the system.

1 Introduction

As biological research is becoming increasingly data driven, scientists are conducting experiments using the cyberinfrastructure (*in silico* experiments) to gather information in public online databases and to test their hypotheses. These heterogeneous, independently developed data sources make traditional approaches insufficient for this type of research and experimentation. Complex queries against several of these databases may provide valuable new insights, but interoperability problems make this difficult. The researcher must often manually cut and paste data from one database resource to another and repeatedly use multiple tools to format and analyze the data, a process that

may take days or weeks. In many investigations, the process stops once the scientist requires a workflow that is not feasible using manual retrieval and analysis.

There is a demand for a methodology that frees users from having to locate the data sources, interact with each data source, and manually combine data in multiple formats from multiple sources. A promising solution to achieve the seamless interoperability among these data sources and analysis tools relies on the emerging technology of service-oriented architecture (SOA). SOA has been recognized as an approach to achieve interoperability among multiple data sources during the past few years [22] [23]. Many large bioinformatics database providers, such as NCBI, EMBL, DDBJ, already make their databases available via a SOA. Emerging toolkits and platforms, such as Soaplab [20] enable many data analysis tools to be wrapped as web services. These existing services permit software engineers to build unified interfaces for scientists to access heterogeneous data sources. The platform independent feature of SOA makes it a feasible solution to integrate increasingly available data analysis tools.

While there are protocols, toolkits, and middleware that are increasingly available to address the majority of technical issues in building a data integration and data analysis environment, the question of how real world problems can be solved successfully using these technologies needs to be answered through practical implementations in a real world context. In this paper, we describe the design and implementation of a web-based data integration and analysis environment. The underlying infrastructure is built upon current web service technologies and bioinformatics middleware to enable biologists to better utilize heterogeneous genomic data. The first prototype of the system is used in a phylogenetic research application, the Mother of Green (MoG). MoG is a collaborative research project on plastid phylogenetic analysis involving information technolo-

gists and biologists. Genomic sequence data is accumulating faster than scientists can find and analyze it using manual search tools. The SOA-based platform allows scientists to extract data and analyze phylogenetic comparisons automatically. The web-based environment enables scientists to more effectively define a task, perform the task at a desired time, monitor the execution status, and view the results. The overall aim of this project is to provide an easy-to-use environment for biologists to research the puzzle of plastid phylogeny and to answer an open question on the phylogenetic history of the plastid genome.

In the rest of this paper, we briefly review web service technologies and related work followed by an overview of the MoG project and a description of the overall system architecture. We then describe a prototype implementation of the system, related issues, and extensions of the system.

2 Related work

The service-oriented architecture (SOA) was proposed initially as an emerging paradigm for business process integration inside or across organization boundaries. It is gaining significant attention from the scientific research community for use in building e-science infrastructures. The proposed standard in grid computing, Open Grid Service Architecture (OGSA) [15], is built upon service-oriented architecture and demonstrates the convergence of the Grid with SOA. Three basic standards in SOA, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI), are sufficient for providing simple atomic services. However, single atomic services are not adequate for developing complex applications. One of the most important feature of SOA is that services developed in different groups can be combined as a workflow to solve complicated problems. This feature leads to several research issues and challenges including service discovery, services composition, and service enactment. Semantic web technology [13] [2] and peer-to-peer technology are used in SOA to automate the service discovery process and make the service enactment more reliable.

BioMOBY is an open source research project which aims to generate an architecture for the discovery and distribution of biological data through web services [26]. The architecture provides a set of foundational functions that allows client programs to expand on the specification to include additional new features. There are two development tracks with different architectures, MOBY-Services (MOBY-S) and Semantic-MOBY (S-MOBY). REMORA [5] is a web server implementation base on the BioMOBY service specification. It provides life science researchers with an easy-to-use workflow generator and launcher, a repository of predefined workflows

and a survey system.

Another project, myGrid, provides e-Science application developers a toolkit based upon a high-level middleware layer. It builds on and extends the Grid framework of distributed computing through a SOA. It not only provides a semantic based service discovery system but also the Taverna workflow bench [17], personalized data repositories, provenance and update notification. The direct users of myGrid are users who build applications using the myGrid toolkit [24]. Bioinformaticians, tool builders and service providers can collectively or selectively employ these middleware services to produce applications that support research in the biological and life sciences [8].

The IRIS [18] project is another active project that targets the service discovery, composition, and interoperability of services required within *in silico* experiments. The IRIS project handles this problem through a semi-automatic procedure for identifying and placing customizable adapters into workflows built by service composition.

Web Service for Bioinformatic Analysis Workflow (Ws-BAW) [27] and Bioinformatic Workflow Builder Interface (BioWBI) [3] are two projects provided by IBM aphaWorks to allow life science researchers to build and execute bioinformatics workflows and share their analysis processes.

3 Motivation

The motivating application is the phylogenomics of the plastid. Named the Mother of Green (MoG) project by an multidisciplinary team of computer scientists and biologists, MoG aims to identify the most recent common ancestor of all plastids. While many biologists support the view that all plastids are descended from a single endosymbiont ancestor, the data are not conclusive due to the missing information and inefficient use of existing information. Using the nucleotide and amino sequences of expressed genes to infer ancient ancestral relationships, MoG investigators hope to identify which of the ancestral plastid genes have traveled into the host nucleus and why some genes are more likely to be transferred than others. The rate of data accumulation, the rapid development of new phylogenetic analysis tools, and the refinement of existing tools simply overwhelm the researchers. The biologists need a better approach than manual or ad-hoc scripting to accumulate and analyze enough relevant data to rigorously test the single ancestor hypothesis.

3.1 Use case

A typical phylogenetic analysis process consisting of multiple manual data collection and data analysis steps is described below and shown in Figure 1.

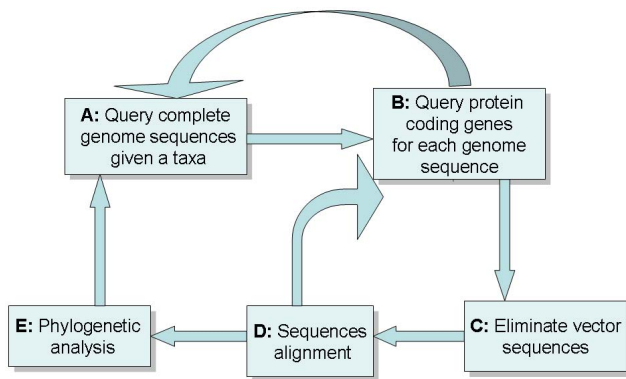


Figure 1. A manual phylogenetic data collection and data analysis process

A) Biologists send a query to a data provider, NCBI for example, through a web-based interface to retrieve whole genome sequence of a specified taxa. After recording the query terms and results, the investigator must examine the list of sequences, delete inappropriate entries and then add new entries based on their knowledge of plastid phylogenomics or from sequences generated in their own lab.

B) For each whole genome sequence, biologists need to find specific protein coding genes, or the specific subunits of protein coding genes, or specific active sites within a specific gene or subunit. This is an iterative process for each entry in the list.

C) Each nucleotide sequence must be checked for vector sequences, a common contaminant of nucleotide sequences in unvetted public databases, and any detected vector contaminants removed.

D) Biologists then choose a subset of these genes and use a sequence alignment program, (e.g. ClustalW), to align the sequences. After viewing the results, biologists may decide to choose another subset for sequence alignment analysis or continue the comparison using phylogenetic tree building tools.

E) Once the initial sequence alignment results prove satisfactory, biologists convert the alignment output to the appropriate data format required by the phylogenetic analysis programs, such as PAUP or Phylip.

3.2 Operational barriers

The data retrieval and data analysis processes need to be repeated multiple times, as different hypothesis are evaluated and new data pours into the public databases. From an operational perspective, this repetition makes the research process time consuming or even impossible using manual approaches. Other barriers also make this particular scientific research process even more difficult.

- *Data collection* The capabilities offered by a data retrieval system cannot always meet the requirements of scientists. Entrez [7] is a web-based data retrieval system available from NCBI that provides integrated access to multiple databases covering a variety of data domains, including complete genomes, nucleotide and protein sequences, gene sequences, three-dimensional molecular structures, literature, and more. However, sometimes scientists are not able to get desired information with a simple query. For instance, “find all of the subunits for the plastid ATP synthase” requires that the investigator first identify the official protein names of all subunits of which there many (atp alpha, atp beta, atp gamma, atp delta, atp epsilon and so on) for the plastid-specific ATP synthase. The next process is to retrieve these sequences for each new genome and to merge these data with the data previously retrieved.

- *Analysis tool usage* Each data analysis program may have different requirements for input data formats even for programs providing similar functionalities. Correct use of these programs and correct implementation of this workflow relies heavily on the researcher having detailed knowledge and understanding of each tool. A typical work unit might be: “find all of the sequences for atp synthase alpha subunit that are most similar to the atp alpha synthase sequence found in *Prochlorococcus*, align the sequence using clustalW, save that output, then reformat the data and submit the sequences to Phylip for phylogenetic analysis”. The output from one data analysis program needs to be fed into the next program as its input with appropriate conversion to the required data format. The rapid development of new data analysis tools and the refinement of existing tools make the manual data conversion process even more difficult.

- *Experimental record keeping* Accurate recording of an *in silico* investigation, including materials, methods, and results is as important as accurate recording of bench top or field experiments. Keeping the provenance data, including the input, output, and intermediate data sets is also critical. Manual organization of these metadata quickly approaches impossibility for anything but the most trivial of queries.

An easy-to-use environment is essential and necessary to support the automation of deep phylogenetic analysis. For many years the data were sparse. Now mountains of data exist but our limited 20th Century tools do not properly equip us to mine for the gems within them. Automation has become necessary.

4 System architecture

The whole system, MoGServ, includes an underlying infrastructure, MoGServ middlelayer, and a web-based environment that provides an easy-to-use interface for scientists to access functions provided by the middlelayer. The system acts as both service consumer and service provider in the context of SOA. While it consumes and aggregates services provided by other service providers, the system also provides services that can be used and integrated by other applications.

There are two roles in the design and implementation of the system, end-users and software developers. End-users are biologists who focus on the study of what information needs to be gathered and what data analysis needs to be performed. The software developers are responsible for several tasks based on end-users requirements: collecting and annotating available services; creating services to implement functions in the specific application; building workflows to automate a variety of tasks required by end-users; providing a flexible, high performance, fault-tolerant infrastructure to execute the workflows; providing a mechanism for end-users to keep track of the origin of the data (data provenance); and providing end-users a web interface to configure a task, monitor the execution status, and view results. An overview of the MoGServ system architecture is given in Figure 2.

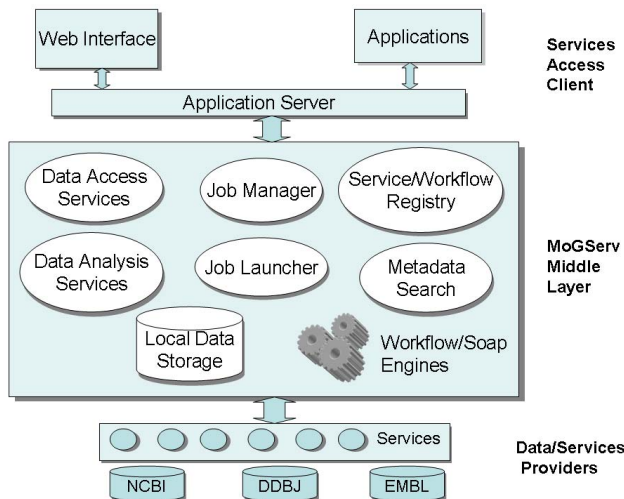


Figure 2. MoGServ System architecture

4.1 Data storage and access service

Data collection from multiple distributed data resources is one of the first steps of a bioinformatics research project. In the MoG project, an *in silico* experiment involves the collection of large data sets, a computational and memory

intensive process that involves daily checking for new information and quality control for each new sequence detected. Some data service providers limit the number of connections to their data server for performance concerns. The refresh rate of the data in a data source is much lower than the rate of end-user requests for the data. Therefore, a local data storage is required to store biological data collected from remote data providers, to avoid repeated vetting of the same data and to insure access to the data for time sensitive projects. The biological data from remote data sources is gathered, aggregated, and integrated into the local database through a set of data access services.

An *in silico* experiment also requires the integration of results from numerous data analysis tools. Recording the intermediate data in the local database allows MoGServ to preserve the data provenance and provides opportunity for end-users to keep track of where a piece of data has come from. The information stored in the local database can be accessed through a set of data access services.

4.2 Service and workflow registry

A service and workflow registry provides a repository to store descriptions of services and workflows that may be used in a phylogenetic study. These services and workflows include both locally constructed and preexisting services. The registry also provides functions to allow inquiries about services or workflows. In the first prototype, neither UDDI-based registry nor semantic-based descriptions are employed. While a UDDI type registry is more business-oriented and may not be a perfect fit for this application, the semantic-based description takes more time to define a common used ontology. The current registry is a simple table with focus on capturing both functional and non-functional properties of services and workflows to support service selections, service and workflow enactment, and provenance data representation. Semantic-based description and inquiry provides the attractive capability of automating service discovery and will be used in the next version of MoGServ. The description of a service or workflow includes attributes as shown in Table 1.

When end-users view results from their experiments, they may ask a question “which algorithm was used to generated the data and what is the source of the data?” Service consumers may prefer a service or workflow based on their preference for a particular algorithm or provider. For example, a sequence alignment service can be implemented using the Sequence Alignment and Modeling System (SAM) or ClustalW.

Table 1. Services and Workflows description

Attributes	Description
<i>id</i>	a unique sequence number assigned to the service/workflow during the registration process
<i>name</i>	the name of the service or workflow
<i>text description</i>	description of the functions provided by the service or workflow
<i>location</i>	the URL of the definition of the workflow or WSDL location of the service
<i>input/output</i>	description of input/output parameters
<i>provider</i>	the name of the service or workflow provider
<i>version</i>	the version of the service or workflow implementation
<i>algorithm</i>	the algorithm used in the service or workflow implementation
<i>invocation method</i>	the method used to execute the service or workflow

4.3 Indexing and querying metadata

The data is best managed with a relational database, however, for searching purposes, an indexer is more efficient. We identify and extract metadata about additional actual data sequences, experiments, services and workflow descriptions in the local database. For example, the metadata of a gene sequence includes *gid*, *accession number*, *name of the sequence*, *from which organism*, and *taxonomy*. An experiment can generate results that leads to new or more detailed information requirements and a new series of experiments. End-users may need to know the origin of a piece of data – “which query was used to get this subset of sequence, when was the data generated, what process was used to generate the results”. This may lead to new experiments using different data sets or even different methods.

These metadata are extracted and indexed by a metadata indexing service. This service is triggered when new data is added into the database. A metadata searching service provides functions to query an index.

4.4 Service and workflow enactment

The system supports both synchronized invocation and asynchronous invocation methods. Synchronized invocation is mostly used for invocation of services or workflows with short running time, e.g. querying sequence data or job information in a local database.

Asynchronized invocation is used for executing long running services and workflows. As shown in Figure 3, the job manager accepts the input parameter of the service/workflow, service/workflow id, and timer. The definition of the services and workflows is found in the registry. A job definition including the services or workflows URL, input parameter, timer, and other metadata of the job information (such as when and who submitted this job) is stored in the database. A job id used to identify the job is generated. The job launcher periodically checks the database to retrieve a service and workflow which needs to be executed at a time point.

Multiple workflow engines are deployed on different nodes to prevent single engine failure and achieve higher performance. A similar mechanism is used for deploying long running services to prevent service failure. Each node hosts a service that is responsible for returning the current load information of the node. This information is used by the job launcher to dispatch a job to an optimal node. With the SOA, it is easy to distribute and invoke workflows and services remotely.

The execution status of the workflow or service is recorded into the database as an attribute of a job description. This information can be used for implementing failure recovery functions, such as restart. The job information accessed through data access services allows end-users to monitor the execution and view the results.

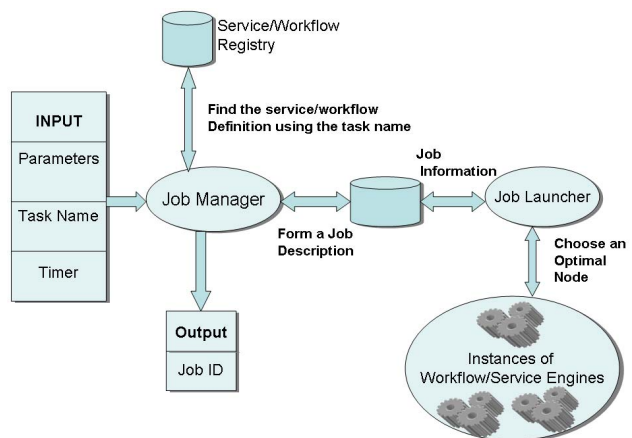


Figure 3. Asynchronized services and workflow invocation model

5 Implementation

5.1 Development and deployment tools

Among a large number of programming platforms for web services development and deployment, Microsoft's

.NET and Sun's J2EE typically are two main choices for applications and middleware developers. With consideration of future extension of the system as well as our previous experience with Java, the J2EE based platform appeared more suitable for MoGServ. In particular, Apache's open source tools - Tomcat(5.0.18) and Axis(1.2RC2), are used.

Tomcat/Axis are active projects with support from the open source community. Another open source software tool, Eclipse, is used to develop the web interface for the system.

There are more than a dozen proposed languages to coordinate messaging and transactions among independent web services. The business process execution language for web services (BPEL4WS) is a promising workflow language since it has wide support from IBM, Microsoft, and BEA. Several workflow enactment engines, such as BPWS4J, Collax, ActiveBPEL, are already in place to support the execution of workflow. While a business-oriented workflow language and corresponding execution engine can be used in the scientific domain [6], the Taverna [17] project possesses more attractive features and naturally fits the development of our system. The Taverna project is open source and a part of the myGrid project developed in the e-Science community to support data-intensive *in silico* bioinformatics experiments. The Taverna workbench provides a graphical tool for building, editing, and browsing workflows and generates a XML-based Simple conceptual unified flow language (Scufl) document. The embedded workflow execution engine, Freefluo, facilitates the testing during the development process. Freefluo, a Java workflow enactment engine which supports the Scufl specification, coordinates execution of the parallel and sequential activities in the workflow and supports data iteration and nested workflows. The enactor can invoke arbitrary WSDL type service operations as well as more specific bioinformatics service operations such as Soaplab and BioMoby.

Apache Lucene [12] is used in our system for building a search engine to support full-text search on sequence data, intermediate data results, and job information stored in the local database. Since Lucene is a search engine library written entirely in Java instead of a command line toolkit, it provides flexibility to write a variety of applications with rich search capabilities.

PostgreSQL(8.0) is used to store all the intermediate data results, job information, sequence data, and services/workflow descriptions.

5.2 Services provision

We create web services using the RPC style due to its easy implementation with full support from most tools. As most bioinformatics applications take a number of input parameters and produce a number of outputs, we use an

XML document to represent the input/output of a service for which a large number of parameters are needed. The XML document is provided as a single input parameter to the service or workflow and the output results are produced as a single XML document. Using this method, the service consumers themselves create a valid and accurate XML document for input while service providers parse the XML and extract the input parameters.

Multiple services are created and deployed on the Tomcat/Axis server using the Java2WSDL and WSDL2Java toolkits. Individual services can be invoked statically or dynamically through a client side application. They can also be used as a building block in the workflow creation process. We separated services provided in the first prototype into the following categories.

Data collection The original data source is NCBI. NCBI's Entrez Programming Utilities (eUtils) provide access to Entrez data outside of the regular web query interface and help for retrieving search results for future use in other environments. With the eUtils SOAP interface, we create services to get data, such as complete genome sequences and specific genes of interest.

Query local database All the intermediate data and job information are stored in the local database to help biologists keep track of the data provenance and monitor the job execution. Also in this particular application, biologists are interested in selecting sequence subsets from the local database and using sequence alignment services to do preliminary comparisons. A set of services are implemented to query desired information.

Indexing and querying metadata The creation and update of each of these indices is done by a service operation. The index service is triggered whenever new data is stored in the database. The query service accepts a query string and an index name to search the index and return output.

Data format services Each particular data analysis tool used in bioinformatics study requires a specific data format as input. A set of data format services in the system is implemented to convert data into an appropriate format. This type of service can be used in a workflow creation process or used explicitly.

Data analysis services Many existing data analysis tools in bioinformatics research are available as command line applications. The creation of a data analysis service is a process to wrap these toolkits as web services. JLaunch [10] is a light-weight Java library for launching command line applications from Java programs. With the JLaunch library, we can write Java programs to execute any type of command line programs.

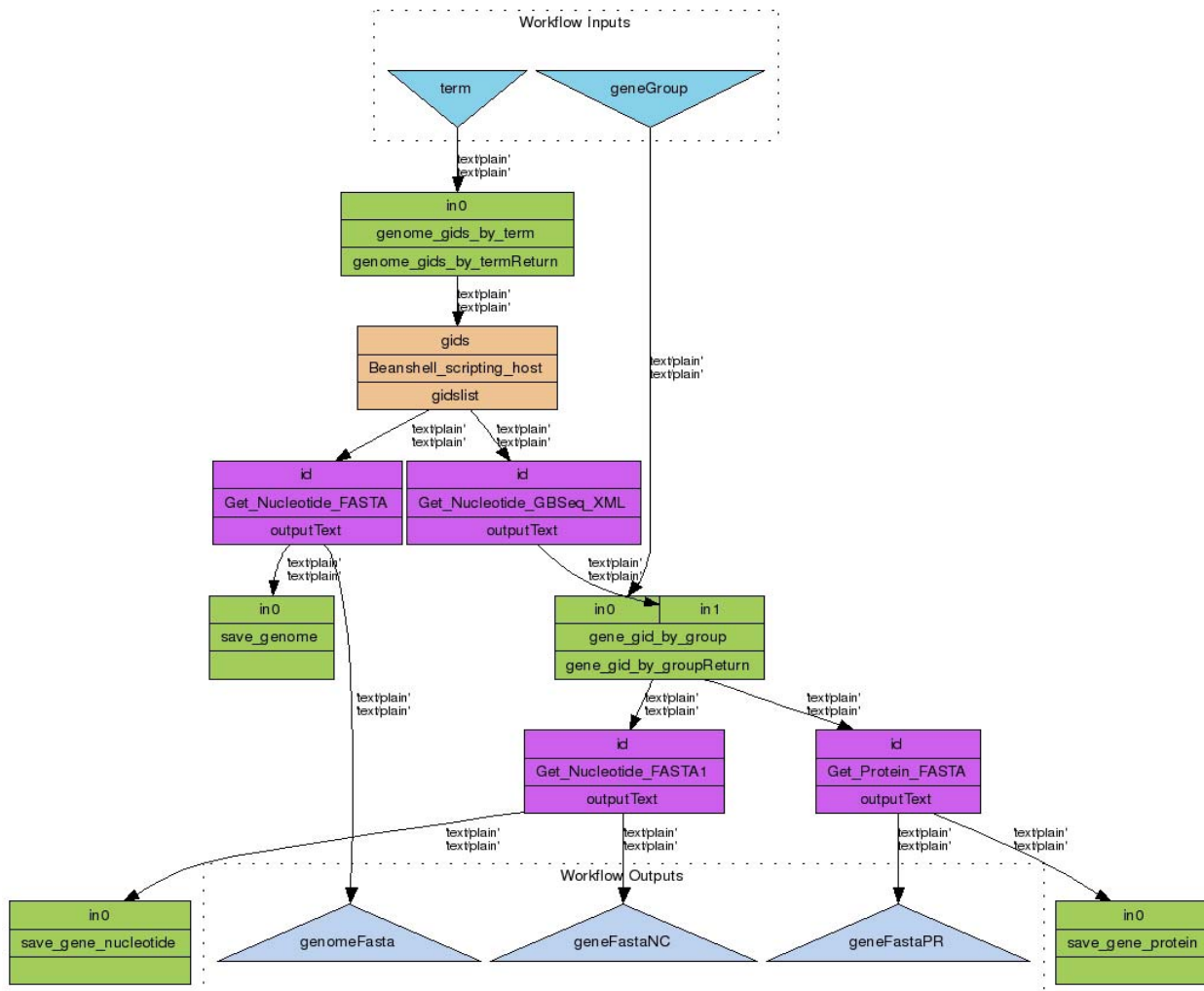


Figure 4. A workflow built using Taverna workbench to get complete genome sequences and specific gene sequences

5.3 Building workflows

A Scuff workflow represents a procedure as a set of processes and the relationships between these processes. Our workflow design uses available services as building blocks whenever possible and creates new ones when necessary. The Taverna workbench provides a graphical tool to build and test workflow as well as a number of integrated bioinformatics services. The Scuff language has some useful features such as implicit iteration and conditional branching that are most important for building workflows in this application. During the construction of workflows, we often encounter the case that output of one service can not be completely fit to the input of the next chosen service.

One approach we take is to create a new service, such as the type of data format service described above, and expose it in the same way as other services. An alternative approach provided in the Taverna workbench is to use the Beanshell scripts [1] to convert the output to appropriate input. We create a number of workflows using Taverna workbench to support the research. One example is shown in Figure 4. It is a workflow used to retrieve complete genome sequence and particular gene sequences from the NCBI site. The workflow accepts two inputs, the query term and the particular gene group. The service `genome_gids_by_terms` returns a String of gids and the Beanshell scripts converts the String to a list of gids. The service `Get_Nucleotide_Fasta`, a third party service,

accepts a gid and returns a sequence in fasta format. The implicit iteration method in the Xscuff workflow enables iteration for all the gids in the list. With the service-oriented architecture, the same services can be used for different workflows, minimizing the need to create new services.

5.4 Web interface

The web interface provides scientists a convenient interface to configure their tasks, monitor the job execution status and view results. It is implemented with a number of server side JSPs (Java Server Pages). The returned results are transformed with appropriate XSLT to HTML pages.

6 Discussion

Although current development and deployment tools haven't implemented all the features claimed in the service-oriented architecture specification, they are actively evolving to make it happen. In particular, the Apache Tomcat/Axis, Taverna workbench, and Freefluo engine enabled the implementation of our first prototype.

In general, SOA offers considerable benefits for building the system: 1) The loose coupled feature of SOA facilitates the distribution of computational intensive processes across multiple nodes; 2) The platform independent feature of SOA facilitates the integration of data from heterogeneous data resources through distributed web services; 3) The composition-of-services feature allows reuse of a service in multiple workflows minimizing the need to create new services; 4) SOA also provides flexibility for building the front-end web application with different languages, e.g. Perl, and deploying on different web service, e.g. Apache/SOAP::lite.

While we believe a simple SOA architecture is appropriate in the design and implementation of our system, there are various aspects of the system that need to be improved. We summarize issues and the directions to enhance the system in this section.

6.1 Issues with the first prototype

Although security was not our major concern during the first prototype implementation, it is an important component in the next implementation. Services and workflows provided in the system allow users to access the computational and data resources in the system with no restriction. A certain level of security is required to prevent abuse of the system and to protect sensitive data and analysis results. An authorization component should be built in the system to enable users to access the permitted services and to personalize their own workspace. A web portal will be built to enable users to create an account, login and logout with

username and password. The user account information including the access level will be stored in a database. The GridSphere portal framework [9], an open-source portlet based web portal, is one of the candidates.

In the first prototype implementation, the same development group acts as both service provider (services/workflow creation) and service consumer (building the web-based application using these services and workflows) roles. While there is no demand for supporting the selection of appropriate services/workflows, the major capability of the index-based services/workflows registry is to keep track of data provenance and to provide definition for performing services/workflows.

However, the index-based syntactic description services/workflows provide limited flexibility for third party service consumers to choose appropriate services/workflows provided in the system and to integrate them into their application without prior knowledge.

The workflow or service execution may fail at some point due to the failure of the enactment engine, failure of the service, and failure of the network fabric [16]. Our system handles these failures during the static workflow design stage and services or workflows invocation stage.

Multiple workflow engines and long running services are deployed on different physical locations. It allows a submitted task to be invoked on the most idle site to achieve higher performance. More importantly, this approach can prevent dispatching services/workflows to the engine with a physical failure. Recording execution status of long running services/workflows in the database allows us to add policies for determining if a failed service/workflow should be restarted. The Taverna workbench and Xscuff provide a capability that allows users to specify an alternate service and to configure basic fault tolerance mechanism during the workflows design time, which can prevent the failure of services at a certain level.

Another more promising, yet more complicated approach for failure recovery is to support the dynamic selection of alternate service during execution time. However, the implementation of this feature requires services to be described in rich semantic formats using a widely accepted ontology.

In the system, the metadata description of sequence, job information, and services/workflows are stored in the database. A set of indexing and querying services allows end-users to trace the origin of the data, which is a desired feature for scientists. Also the workflow engine and Xscuff provides mechanisms to record more detailed information including the type of processor, status, start and end time, and a description of the service operation. A systems administrator may be interested in using this information to investigate how results, in particular erroneous or unexpected ones, were produced by workflow processes.

6.2 Extension of the system

Although the first prototype of the system focuses on design and implementation based on relatively mature technologies in service-oriented architecture, we are extending the system to address some issues described above with grid computing and semantic web technologies.

Grid technologies specify the mechanisms for distributed resource management, coordinated fail-over, and security. As the Grid technologies and Grid framework Globus toolkit [25] in particular are evolving towards the OGSA standard, integration of the Grid technologies to the system can help address some issues discussed above. The convergence of service-oriented architecture and Grid technology allow us to enhance the system through the integration of existing components.

In a scientific domain, the process used to generate the output of a service and workflow is often as important as the result. As is the case with bench scientists, *in silico* investigators will decide for themselves which methods and which data will be used for their study as well as what kind of outputs they are expecting. In the first prototype implementation, this requirement is satisfied through close collaboration among team members.

As this system will be used by a phylogenomics research community that spans multiple disciplines, different investigators will have their own methods for approaching problems of common interest. A mechanism that allows end-users to define the workflow at a higher level of abstraction is required. Instead of choosing specific services to form a workflow, scientists would rather define a workflow by specifying functions that a service should provide. Different levels of training and experience also require different levels of abstraction. For example, a graduate student in a particular research domain may have limited knowledge of the methods available to perform an experiment, while an experienced investigator may know ahead of time which building blocks are required and which approach is most efficient for the scientific hypothesis to be tested. We represent different abstraction levels in Figure 5. End-users may need to define the workflow at any one of these four stages based on their knowledge of provided services.

A concrete workflow, which can be sent to a workflow engine, is represented at the fourth phase. The conversion from the third phase to the fourth phase is related to choosing an instance of a service with Quality of Service (QoS) metrics. One service interface may have multiple implementations provided by different service providers. These implementations have different quality properties such as trustworthiness, cost, execution time, and so on. An optimal service should be chosen during this conversion process. The conversion from the second phase to the third phase requires mapping a particular task to a service, or a

sequence of multiple services.

This mapping process can be accomplished manually by software developers in an ad-hoc way, like the approach we took in the implementation of the first prototype. This approach relies heavily on developers' knowledge of services and logical ordering in the workflow.

Preferably, this process should be able to be done partially or wholly automatically. In order to support this semi-automatic or automatic process, a complete presentation of knowledge should be in place to allow software agents to substitute the work of the human. Using semantic web technology, in particular OWL and OWL-S, to represent the ontological representation of domain knowledge and semantic description of services is a promising approach. Semantic web technology offers promising features for supporting bioinformatics research [4]. Some bioinformatics middleware, such as the myGrid and BioMoby projects, have their own approaches to support automated discovery and composition of services using semantic web technology [11]. Much research has been done exploring AI planning techniques for automation of the composition process. The long term goal of a successful composition mechanism should meet several requirements: connectivity, quality of service, correctness, and scalability [14].

Although there are still practical difficulties in developing semantic web services, we believe that the appearance of tools for creating ontologies, annotating services [21], and development of widely accepted domain ontologies allow us to add semantics into our system and support the automation of the mapping process.

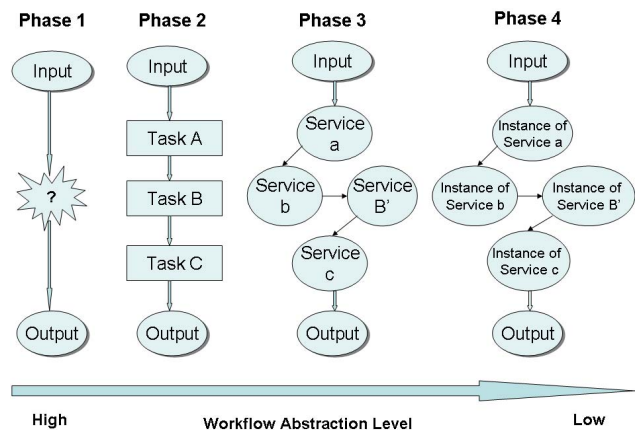


Figure 5. Abstraction of user defined workflows

7 Conclusion

As both data and tool providers begin to present their resources with web services interfaces, and as open source tools and middleware for supporting web services, workflow generation, and enactment become more available, biologists will begin to use those available services, as well as begin to provide service access to their databases and programs for sharing within the bioinformatic community [17]. Our system is a demonstration of progress toward this goal.

In summary, current SOA standards and toolkits are sufficient to build the first prototype of MoGServ. MoGServ is in its early stage of development with limited services and workflows available. The basic implemented functionalities enable the user to collect data and do preliminary data analysis as well as metadata searching. By using the system, scientists are able to get some scientific insights about the alpha subunit of ATP synthase and indicate that it retains the signal of a very ancient line of descent while having enough polymorphism to infer phylogenetic relationships [19].

Building the system upon the SOA provides us flexibilities to integrate services, to build a variety of workflows, and to build a web portal for scientists to access the system via a web interface. New features and services are continuously being added to the system in response to scientists' feedback and requirements. The future direction of our research will be to focus on enhancing the system using semantic web and grid computing technologies.

References

- [1] Light weight scripts for java. <http://www.beanshell.org/>.
- [2] T. Berners-Lee, J. Hedler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
- [3] Bioinformatic workflow builder interface (biowbi). <http://www.alphaworks.ibm.com/tech/biowbi>.
- [4] D. Buttler, M. Coleman, T. Critchlow, R. Fileto, W. Han, C. Pu, D. Rocco, and L. Xiong. Querying multiple bioinformatics information sources: Can semantic web research help? *SIGMOD Record*, 31(4):59–64, 2002.
- [5] S. Carrere and J. Gouzy. Remora: a pilot in the ocean of biomoby web-services. *Bioinformatics*, 22(7), 2006.
- [6] R. de Knikker, Y. Guo, J. long Li, A. K. Kwan, K. Y. Yip, D. W. Cheung, and K.-H. Cheung. A web services choreography scenario for interoperating bioinformatics applications. *BMC Bioinformatics*, 5(25), 2004.
- [7] C. Goble, C. Wroe, R. Stevens, and the myGrid consortium. The mygrid project: services, architecture and demonstrator. In *UK e-Science AHM*, September 2003.
- [8] Gridsphere portal framework. <http://www.gridisphere.org/gridsphere/gridsphere?cid=2>.
- [9] Jlauch from duke bioinformatics shared resource. <http://dbsr.duke.edu/>.
- [10] P. Lord, S. Bechhofer, M. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *Third International Semantics Web Conference (ISWC2004)*, 2004.
- [11] Apache lucene. <http://lucene.apache.org/java/docs/index.html>.
- [12] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, pages 46–53, March/April 2001.
- [13] N. Milanovic and M. Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51–59, November/December 2004.
- [14] Entrez: Making use of its power. *Briefings in bioinformatics*, 4(2), June 2003. <http://www.ncbi.nih.gov/>.
- [15] Links to open grid service architecture. <http://www.globus.org/ogsa/>.
- [16] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, A. Wipat, and P. Li. Taverna, lessons in creating a workflow environment for the life sciences. In *GGF workflow workshop*, 2004.
- [17] 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.
- [18] U. Radetzki, U. Leser, S. C. Schulze-Rauschenbach, J. Zimmermann, J. Lussem, T. Bode, and A. B. Cremers. Adapters, shims, and glue—service interoperability for in silico experiments. *Bioinformatics*, 22(9), 2006.
- [19] J. Romero-Severson. Use case: How mog web services enable scientific discovery. Technical report, University of Notre Dame, August 2006. <http://www.nd.edu/mog/Papers/papers.html>.
- [20] Soap-based analysis web service developed in the european bioinformatics institute (ebi). <http://www.ebi.ac.uk/soaplab/>.
- [21] N. Srinivasan, M. Paolucci, and K. Sycara. Semantic web service discovery in the owl-s ide. In *Proceeding of the 39th Hawaii International Conference on System Sciences*, 2006.
- [22] L. Stein. Creating a bioinformatics nation. *Nature*, 417(9), 2002.
- [23] L. D. Stein. Integrating biological databases. *Nature Reviews genetics*, 4, 2003.
- [24] R. Stevens, K. Glover, C. Greenhalgh, C. Jennings, S. Pearce, P. Li, M. Radenkovic, and A. Wipat. Performing in silico experiments on the grid: A users perspective. In *Proc UK e-Science programme All Hands Conference*, 2003.
- [25] The globus project. <http://www.globus.org>.
- [26] M. D. Wilkinson and M. Links. Biomoby: An open source biological web service proposal. *Briefings in bioinformatics*, 3(4), 2002.
- [27] Bioinformatic analysis workflow (wsbaw). <http://www.alphaworks.ibm.com/tech/wsbaw>.