Bioworks: A Workflow System for Automation of Bioinformatics Analysis Processes

Youngmahn Han

Supercomputing Center, Korea Institute of Science and Technology Information
Daejeon, South Korea
hans@kisti.re.kr

Abstract

Over the past decade, the rapid development of high-throughput technologies has lead to an explosive growth in biological information. Many molecular biology fields involve in-silico experiments using bioinformatics applications. A workflow can be well-defined model for bioinformatics analysis processes to be performed as a chain of interlinked data process tasks. We have developed a client/server-based workflow system called Bioworks, supporting large-scale analysis through the high performance cluster computing resources. In this paper, we highlight implementation methods in Bioworks, for meeting key requirements of effective workflow systems for bioinformatics analysis processes.

Keywords: *bioinformatics; workflow; provenance; data integration; large-scale analysis; automation*

1. Introduction

In the post-genomic era, the development of high-throughput technologies has led to an explosive growth in biological information. Furthermore, emerging Next Generation Sequencing (NGS) technology brought unprecedented volumes of DNA sequence data [1].

Bioinformatics is the application of statistics and computer technology to mange and interpret these massive data. Currently, many molecular biology fields, including genomics, transcriptomics, proteomics, metabolomics, and pharmacogenomics involve *in-silico* experiments using bioinformatics applications.

Many bioinformatics analysis processes are performed as a chain of interlinked data process tasks. A pipeline or "workflow" can be a well-defined model for bioinformatics analysis processes, , with a specific structure defined by the topology of data-flow interdependencies, and a particular functionality arising from the data transformations applied at each step [2]. A typical example for bioinformatics workflow given in [3] describes chained tasks involved in determining the protein family of a protein that corresponds to a given DNA sequence as shown in Figure 1.

Workflow systems emerged as a glue to orchestrate heterogeneous biological information and data analysis tools. They could be considered as a comprehensive approach for dealing with the data explosion. This has led to the recent development of workflow management systems (WMS) in bioinformatics experiments. Catalogs of bioinformatically-capable WMS can be found in recent reviews [4-6].

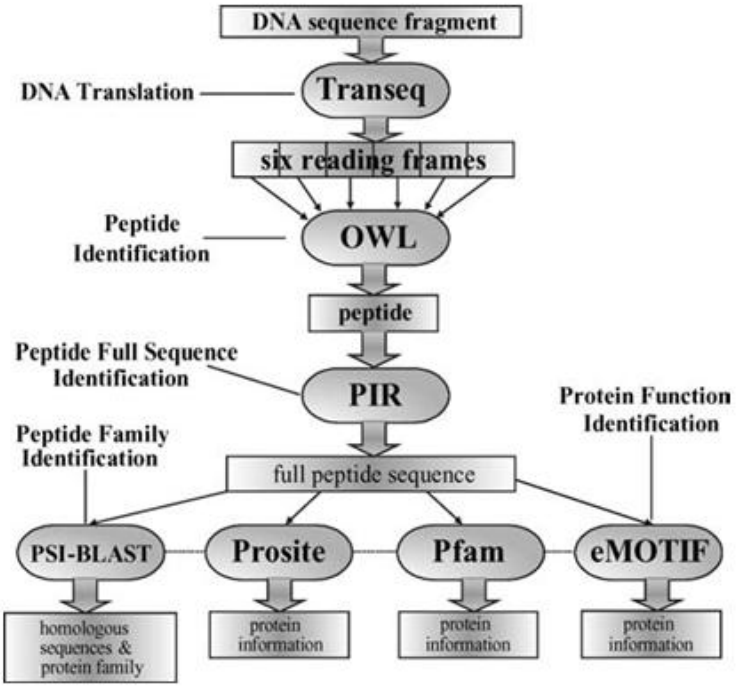

Figure 1. One of bioinformatics workflows for searching for protein family information from the given DNA sequence fragment.

Taverna [7] from EBI is the best-known standalone WMS for bioinformatics experiments. It was developed as a part of the myGrid project. Taverna provides the standardized workflow markup language called SCUFL (Simple Conceptual Unified Flow Language), a Java-based workbench to be able to compose complex workflows from unit tasks accessing both remote and local processors of various kinds, to launch execution of workflows and to display different types of results, including text, web pages and various kinds of images and diagrams. Taverna also provides special local processors to allow for executing Java-based dynamic scripts and R scripts. The web portal called myExperiment provides useful environments to share Tarverna's workflows.

Biopipe [8] is a workflow framework or software library which allows researchers to create pipelines for bioinformatics analysis pipelines based on Perl and BioPerl, to execute pipelines on clusters. Biopipe currently does not provide user-friendly interfaces for compose pipelines. It rather aims to allow researchers to focus on protocol design to plug in different Perl modules for easy bioinformatics pipelines.

Triana [9] is a problem solving environment developed at Cardiff University that providing graphical interfaces to enable the composition of scientific applications. Triana environment is designed as a series of pluggable components, which can easily be integrated with other systems.

Kepler [10] is one more scientific WMS based on the Ptolemy II system [R47] developed by the Electrical Engineering community at UC Berkeley for circuit design and simulation and has been used in various scientific domains, including bioinformatics, computational chemistry. Kepler adopts actor-oriented modeling for workflows. A workflow is considered as a composition of independent component called actors. Communication between actors occurs through interfaces called input/output ports. In addition of the ports, actors have

parameters, which configure and customize the behavior. Actors, or more precisely their ports, are connected to one another via channels. The execution of actors is independently triggered by the availability of inputs of upstream actors. Kepler also provides support for Web services and Grid extensions.

BioWMS [11] is a WMS that supports, through a web-based interface, the definition, the execution and the results management of a bioinformatics experiment. BioWMS has been implemented over an agent-based middleware. It dynamically generates, from a user workflow specification, a domain-specific, agent-based workflow engine.

There are mainly required considerations to effectively implement these WMS as follows:

- There are many bioinformatics tools and databases literally developed by geographically distributed organizations, research institutes, or related industries across the world. Some make their tools web accessible; some provide command line based standalone programs or software libraries. Standardization and extensible integration of distributed tools is necessary for providing seamless access to them.
- Bioinformatics tools are highly heterogeneous in their input/output data types. These heterogeneity leads to be difficult to make links among tool tasks according to data flow. A WMS should provide flexible integration methods to resolve data type heterogeneity.
- Workflow scalability is important to help in large-scale data analysis like NGS data analysis, protein interaction network analysis, and docking simulation through high performance computing resources, e.g., running a large number of parallel jobs on a cluster computer. However, most research groups seem to be impossible to maintain such computing resources due to the high cost of computer hardware and the lack of professional human resources to manage and utilize them.
- Reproducibility of scientific analyses and processes is at the core of the scientific method, in that it enables researchers to evaluate the validity of each other's hypothesis and to repeat techniques and analysis methods to obtain scientifically similar results. In order to support reproducibility, WMS should capture and generate provenance information as a critical part of the workflow-generated data. Provenance information can be referred as a historical metadata that provides explanations on how a particular intermediate result data has been generated from the given input data [2].

In order to meet above required considerations, a bioinformatically-capable WMS called Bioworks has been developed under the project funded by KISTI. Bioworks is implemented in Java, and based on client/server architecture adopting Java Web Start technology [12] and Web Services for data transmissions. Bioworks has been tested on JRE (Java Runtime Environment)-installed platforms including Window, Linux and Mac.

Here, we present implementation methods to cope with the above requirements in Bioworks. Next Section 2 introduces the Java-based client/server architecture. Section 3 presents how distributed and assorted tools are integrated and their heterogeneity in tool input/output data types is resolved. Workflow execution model is presented in Section 4. Implementation methods for data management and capturing workflow-based provenance information necessary for reproducibility are presented in Section 5. Section 6 includes a summary of the features of Bioworks and direction for future work.

2. Client-Server Architecture

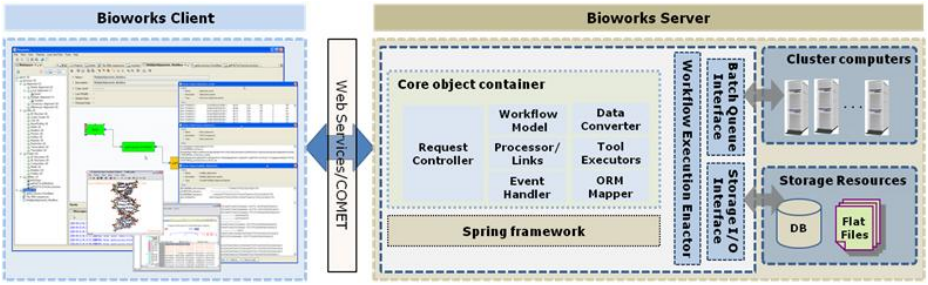

Figure 2. The Client/server-based System Architecture of Bioworks

As shown in Figure 2, Bioworks is based on client/server architecture. Workflows are executed at the server side on high performance cluster computers. As a consequence, this architecture is especially useful for large-scale analysis taking long time, requiring high performance computing. Users can monitor the status of workflow execution through the client program anywhere, anytime. The Bioworks client program provides the user-friendly graphical user interface (GUI) which enables users to easily compose workflows for complex analysis. Especially, by adopting Java Web Start, it can be automatically installed and upgraded via web. The GUI mainly includes Workspace Explorer, User Data Editor, Message Console, and Result Browser as shown in Figure 3:

- Workspace Explorer is a GUI component which is similar with Windows Explorer, which allows users to hierarchically explore and manage their user data including workflows, tools, and text-based biological data such as sequences, alignments, and molecular interaction networks.
- User data Editor provides a tabbed interface which contains different edit interfaces corresponding to each user data. For example, Workflow Editor is a tab item which provides user-friend interfaces to allow users to easily edit their own workflows.
- Message Console allows users to monitor information related to the execution status of their workflows.

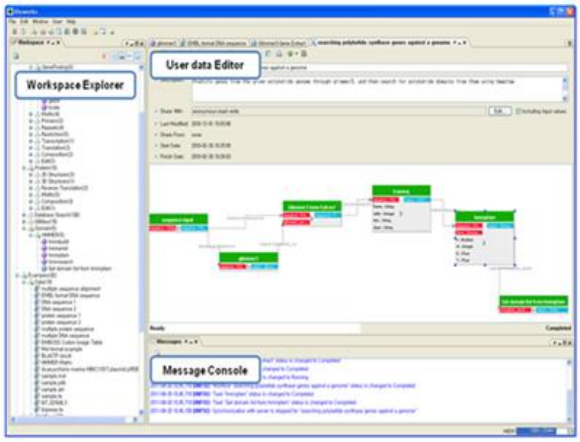

Figure 3. Main GUI components of the Bioworks client program including Workspace Explorer, User data Editor, and Message Console

The workflow composed by the client program is transferred as a Java object model via Web Services and stored to the server. Once the execution of the workflow is requested to the server, the workflow execution enactor is activated, fetches the workflow from the database, and executes the workflow. Intermediate results are stored in the database or file system. The user can fetch each result and inspect them before the entire workflow execution is completed. This is particularly useful for long-running workflows by allowing an early evaluation of results.

In Bioworks, the object data transmission between client/server is archived via Web Services. Web Services machine-oriented network services based on XML, usually communicating by using the Simple Object Architecture Protocol (SOAP). WS have already been implemented by many institutes and service providers in the biomedical field [13]. WS provide a standardized programming interface so that software tools can effectively make access to information and services. This also allows the extensible client application environment, i.e., the client program can be implemented in various programming environments like the Java standalone program, web applications without any change of the server program.

3. Bioinformatics Tool Integration

There are many bioinformatics tools and databases literally developed by geographically distributed organizations, research institutes, or related industries across the world. They make their tools in various access methods including command-line based standalone programs, software libraries, and Web services. Standardization and extensible integration of distributed tools is necessary for providing seamless access to them. Figure 4 shows the class diagram for the integration of tools with several different access methods. *BioTool* is the abstract base class which defines abstract methods like *beforeExecute, execute, afterExecute* to execute the mapped tool. Those methods are implemented in child classes of the *BioTool* including *CommandLineBioTool*, *WebServicesBioTool*, and *ScriptBioTool* for command line tools, Web Services tools, and dynamic script tools respectively.

In order to resolve the heterogeneity in tool input/output data types, an ontology is served as a common data type model for tool input/output format in Bioworks. An ontology is the explicit and formal specification of conceptualization in a given domain of interest. It consists of a set of concepts expressed by using a controlled vocabulary and the relationships among these concepts. Ontologies can add semantic metadata to the resources, improve data accessibility and support interoperability among resources.

Ontologies are widely used for the semantic integration of heterogeneous bioinformatics tools [3, 14-17]. In Bioworks, hierarchical ontology terms like sequences, alignments, protein structures, and molecular interaction networks are provided for common data type models of tool inputs/outputs. During workflow construction, two tools can be linked if the output's ontology term of the upstream tool has 'is a' relationship with the input's ontology term of the downstream tool. The Jena [18] API is used for handling the ontology definition file and validating links.

For type-mismatched link, Bioworks allows users to create a user-defined script for mandatory converting the output data of the upstream task. These scripts can be written by various programming languages including Java, Python, and Ruby in Bioworks. This highly increases the flexibility in composing the workflow from heterogeneous tools.

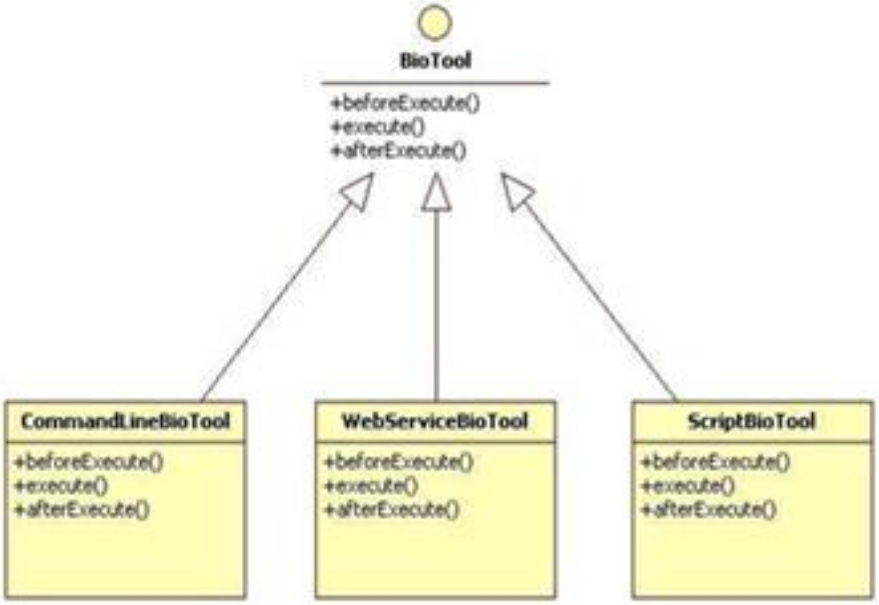

Figure 4. The object-oriented class diagram for the integration of tools with different access methods.

4. Workflow Execution Model

The directed graph, either acyclic (DAG) or the less used cyclic (DCG), is the most common workflow representation. Condor's DAGMan is commonly used DAG format for specifying workflows of dependent jobs. It is the underlying execution workflow representation for Pegasus. Taverna uses an XML-based DAG format called SCUFL. Triana uses a DCG format of its own although it is able to import and export from different formats including DAGMan and the Virtual Data Language [19]. Kepler uses yet another directed graph representation MOML (MOdel Markup Language) inherited from Ptolemy II [20].
In Bioworks, the workflow is represented as the DAG that tasks, links among them are regarded as vertices, edges respectively. Tasks are sequentially executed along the data flow in according with the breadth first search for the graph. The start task becomes the vertex which has no incoming edges. A task can be executed once its upstream tasks are completed; thereby their outputs are set to inputs of the task. The data transfer in a link involves the validation of transition condition and the data conversion. Once each task is executed, the mapped tool is invoked and executed in a variety of ways including command-lines, Web services, and run-time scripts.

For supporting large-scale analysis, command-line tools are executed on the cluster computing resources that are scheduled by the batch queue system, e.g. the Portable Batch System (PBS) as shown in Figure 5.

The workflow execution enactor submits to the batch queue system as a scheduled job for the execution of command-line tools. The batch queue system executes sequentially each job on clustered computing resources. Job statuses and results are synchronized with task statuses and results in the database by the JobMonitor, respectively. This allows users to obtain statuses and intermediate results of workflow's tasks in real-time through the client program.

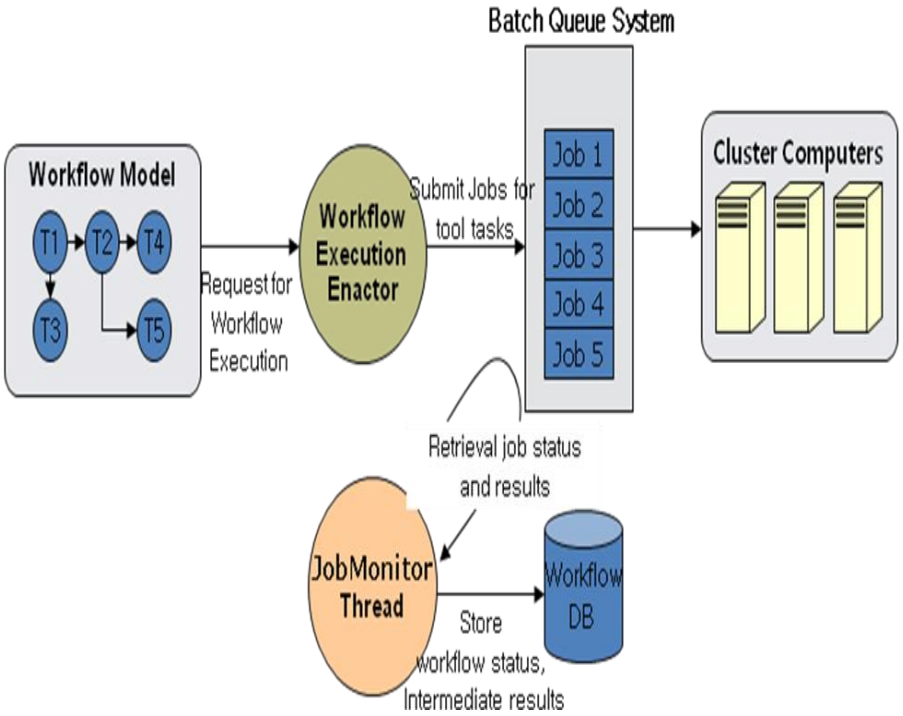

Figure 5. The workflow execution flow supporting large-scale analysis through high performance cluster computers.

User-defined script tools are necessary for making links among heterogeneous tool tasks; for example, it needs to manually implement the tool for obtaining high scored sequences from results of the BLAST similarity search. Bioworks provides the dynamic script launcher that can dynamically execute user-defined scripts developed in various languages including Java, Ruby and Python. The dynamic execution of these user-defined scripts must take care of security issues such as abnormal server shutdown, access to confidential resources. The script launcher uses the SecurityManager provided by the Java Security Architecture to control access of script codes to resources of the server. The Security Manager controls access of operations in client codes to specific computing resources. Once the script is executed under the control of the SecurityManager, access to specific resources is only allowed for operations granted by the security policy that is defined in a standardized text file.

5. Data Management and Provenance

In order to capture workflow-based provenance information, data models related to the workflow including workflow execution metadata, generated results, and input data are historically stored into the database in Bioworks. The object-oriented class diagram for implementation of the workflow-based data models represented as a DAG is shown in Figure 6.

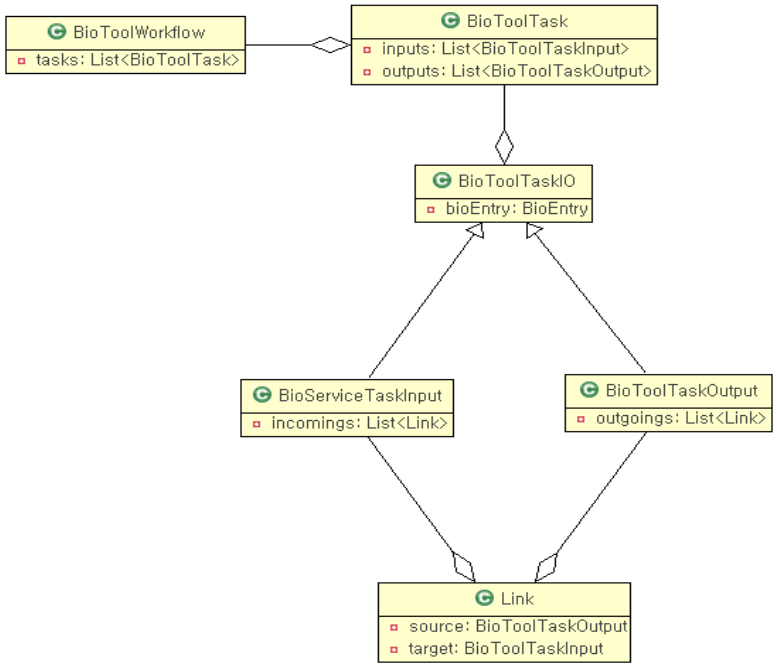

Figure 6. The object-oriented class diagram for workflow data models

Bioworks uses the rational database management system (RDBMS) like MySQL. This leads the inconsistency between object-oriented data models at the application layer and rational data models at the persistence layer; thereby the complexity of the implementation of the whole system is dramatically increased. In order to resolve this inconsistency, several programming techniques like the Object-relation mapping (ORM) have been developed for the automation to convert the object data model to the relational data model and vice versa. In Bioworks, the Java-based ORM engine like Hibernate [21] is used for implementation of the object-oriented persistent layer.

The Hibernate allows for the automation of the ORM through XML-based definitions of object models.

6. Conclusion

Bioinformatics is the application of statistics and computer technology to mange and interpret huge amount of biological data generated from rapidly evolving high-throughput instruments in molecular biology fields. Workflow systems have emerged as a glue to orchestrate heterogeneous biological information and data analysis tools. They could be considered as a comprehensive approach for dealing with the data explosion.

There are commonly required considerations to effectively implement WMS like standardization and extensible integration of distributed tools to provide seamless access to them, flexible integration methods to resolve data type heterogeneity, workflow scalability to support large-scale data analysis like NGS data analysis, and capturing provenance information for the scientific reproducibility. In order to meet these requirements, a bioinformatically-capable WMS called Bioworks has been developed. In this paper, implementation methods in Bioworks to cope with the above requirements were presented. Bioworks is implemented in Java, and based on client/server architecture adopting Java Web

Start technology and Web Services for data transmissions. Bioworks supports large-scale analysis through high performance cluster computers, and provides workflow-based provenance information for the reproducibility and traceability. Future works will focus on enhancing the system performance for supporting large-scale analysis, and providing workflow-based environments for collaborative researches through Bioworks.

Acknowledgment

The author would like to acknowledge the Bioinformatics team of the Supercomputing Center: Seok Jong Yu, Insung Ahn, and Yongseong Cho; and Young-Hwan Bang who has assisted me one way or another. This work is supported by the Korea Institute of Science and Technology Information.

References

[1] Elaine R. Mardis, "The impact of next-generation sequencing technology on genetics", Trends in Genetics, vol. 24(3), pp. 133-141, 2008.

[2] Yolanda Gil, et al., "Examining the Challenges of Scientific Workflows", IEEE Computer, vol. 40, pp. 24-32, 2007.

[3] Malika Mahoui, et al., "A Dynamic Workflow Approach for the Integration of Bioinformatics Services", Cluster Computing, vol. 8, pp. 279–291, 2005.

[4] Paolo Romano, "Automation of in-silico data analysis processes through workflow management systems", Briefings in Bioinformatics, vol. 9, pp. 57-68, 2007.

[5] Abhishek Tiwari, Arvind K.T. Sekhar, "Workflow based framework for life science informatics", Computational Biology and Chemistry, vol. 31, pp. 305-319, 2007.

[6] Ewa Deelman, Dennis Gannon, et al., "Workflows and e-Science: An overview of workflow system features and capabilities", Future Generation Computer Systems, vol. 25, pp. 528-540, 2009.

[7] Oinn T, Addis M, Ferris J, et al. "Taverna: a tool for the composition and enactment of bioinformatics workflows", Bioinformatics, vol. 20: pp. 3045–54, 2004.

[8] Shawn Hoon, Kiran Kumar Ratnapu, Jer-ming Chia, et al., "Biopipe: A Flexible Framework for Protocol-Based Bioinformatics Analysis", Genome Res., vol. 13, pp. 1904-1915, 2003.

[9] Matthew Shields, Ian Taylor, "Programming Scientific and Distributed Workflow with Triana Services, Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems), vol. 18, pp. 1021–37, 2006.

[10] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, et al., "Scientific Workflow Management and the Kepler System", Concurr. Comput.: Pract. Exp., vol. 18, pp. 1039 – 1065, 2005.

[11] Bartocci E, Corradini F, Merelli E, et al., "BioWMS: a web based workflow management system for bioinformatics", BMC Bioinformatics, vol. 8, S2, 2007.

[12] Java WebStart Technology, http://www.oracle.com/technetwork/java/javase/overview-137531.html.

[13] Pieter B. T. Neerincx and Jack A. M. Leunissen, "Evolution of web services in bioinformatics", Briefings in Bioinformatics, vol. 6, pp. 178–188, 2005.

[14] Patricia G. Baker, Carole A. Goble, et al., "An ontology for bioinformatics applications", Bioinformatics, vol. 15, pp. 510-520, 1999.

[15] C. Wroe, R. Stevens, C. Goble, A. Boberts and M. Greenwood, "A suite of DAML + OIL ontologies to describe bioinformatics web services and data", Int. J. Cooperative Inf. Syst., vol. 12 (2), pp. 197-224, 2003.

[16] Romano P, Bartocci E, Bertolini G, et al., "Biowep: a workflow enactment portal for bioinformatics applications", BMC Bioinformatics, vol. 8, S19, 2007.

[17] M.D. Wilkinson and M. Links, "BioMOBY: An open-source biological web services proposal", Briefings in Bioinformatics, vol. 3(4), pp. 331– 341, 2002.

[18] Jena, http://jena.sourceforge.net/

[19] I. Foster, J. Voeckler, M. Wilde, Y. Zhao, “Chimera: A virtual data system for representing, querying, and automating data derivation”, in: 14th International Conference on Scientific and Statistical Database Management, SSDBM'02, IEEE Computer Society Press, New York, pp. 37-46, 2002.

[20] Buck, J., Ha, S., Lee, E.A., Messerschmitt, D.G., “Ptolemy: a framework for simulating and prototyping heterogeneous systems”, Int. J. Comput. Simul., vol. 4, pp. 155–182, 1994.

[21] Hibernate, http://hibernate.org