A Taxonomy of Scientific Workflow Systems for Grid Computing

Jia Yu and Rajkumar Buyya[*]

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
htpp://www.gridbus.org

ABSTRACT
With the advent of Grid and application technologies, scientists and engineers are building more and more complex applications to manage and process large data sets, and execute scientific experiments on distributed resources. Such application scenarios require means for composing and executing complex workflows. Therefore, many efforts have been made towards the development of workflow management systems for Grid computing. In this paper, we propose a taxonomy that characterizes and classifies various approaches for building and executing workflows on Grids. The taxonomy not only highlights the design and engineering similarities and differences of state-of-the-art in Grid workflow systems, but also identifies the areas that need further research.

Keywords
Grid computing, Taxonomy, Scientific workflows.

1. INTRODUCTION
Grids [9] have emerged as a global cyber-infrastructure for the next-generation of e-Science applications, by integrating large-scale, distributed and heterogeneous resources. Scientific communities, such as high-energy physics, gravitational-wave physics, geophysics, astronomy, and bioinformatics, are utilizing Grids to share, manage and process large data sets. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications, and scientific instruments need to be orchestrated while managing workflow operations within Grid environments [15].

Scientific workflow is concerned with the automation of scientific processes in which tasks are structured based on their control and data dependencies. The workflow paradigm for scientific applications on Grids offers several advantages, such as (a) ability to build dynamic applications which orchestrate distributed resources, (b) utilizing resources that are located in a particular domain to increase throughput or reduce execution costs, (c) execution spanning multiple administrative domains to obtain specific processing capabilities, and (d) integration of multiple teams involved in management of different parts of the experiment workflow – thus promoting inter-organizational collaborations.

In the recent past, several Grid workflow systems have been proposed and developed for defining, managing and executing scientific workflows. In order to enhance understanding of the field, we propose a taxonomy that primarily (a) captures architectural styles and (b) identifies design and engineering similarities and differences between them. The taxonomy provides an in-depth understanding of building and executing workflows on Grids. It compares different approaches and also helps users to decide on minimum subset of features required for their systems.

The rest of the paper is organized as follows: Section 2 presents taxonomy that classifies approaches based on major functions and architectural styles of Grid workflow systems. In Section 3, we map the proposed taxonomy onto selected Grid workflow systems and conclude in Section 4.

2. TAXONOMY
The taxonomy characterizes and classifies approaches of scientific workflow systems in the context of Grid computing. It consists of four elements of a Grid workflow management system: (a) workflow design, (b) workflow scheduling, (c) fault tolerance and (d) data movement (see Figure 1). In this section, we look at each element and its taxonomy briefly. A detailed taxonomy and in depth discussion on its mapping can be found in [23].

2.1 Workflow Design
Workflow design determines how workflow components can be defined and composed.

2.1.1 Workflow Structure
A workflow is composed by connecting multiple scientific tasks according to their dependencies. Workflow structure indicates the temporal relationship between there tasks. In general, a workflow can be represented as a *Directed Acyclic Graph (DAG)* or a *non-DAG*.

In DAG-based workflow, workflow structure can be categorized as *sequence*, *parallelism*, and *choice*. Sequence is defined as an ordered series of tasks, with one task starting after a previous task has completed. Parallelism represents tasks which are performed concurrently, rather than serially. In choice structured workflows, a task is selected to execute at run-time when its associated conditions are true. In addition to all structures contained in a DAG-based, a non-DAG workflow also includes *iteration* structure, in which sections of workflow tasks in an iteration block are allowed to be repeated.

[*]Corresponding author, raj@cs.mu.oz.au

2.1.2 Workflow Model/Specification

Workflow Model (also called workflow specification) defines a workflow including its task definition and structure definition. There are two types of workflow models, namely *abstract* and *concrete*.

In the abstract model, a workflow is described in an abstract form, in which the workflow is specified without referring to specific Grid resources for task execution. In contrast, the concrete model binds workflow tasks to specific resources. Given the dynamic nature of the Grid environment, it is more suitable for users to define workflow applications in the abstract model. A full or partial concrete model can be generated just before or during workflow execution, according to the current status of resources.

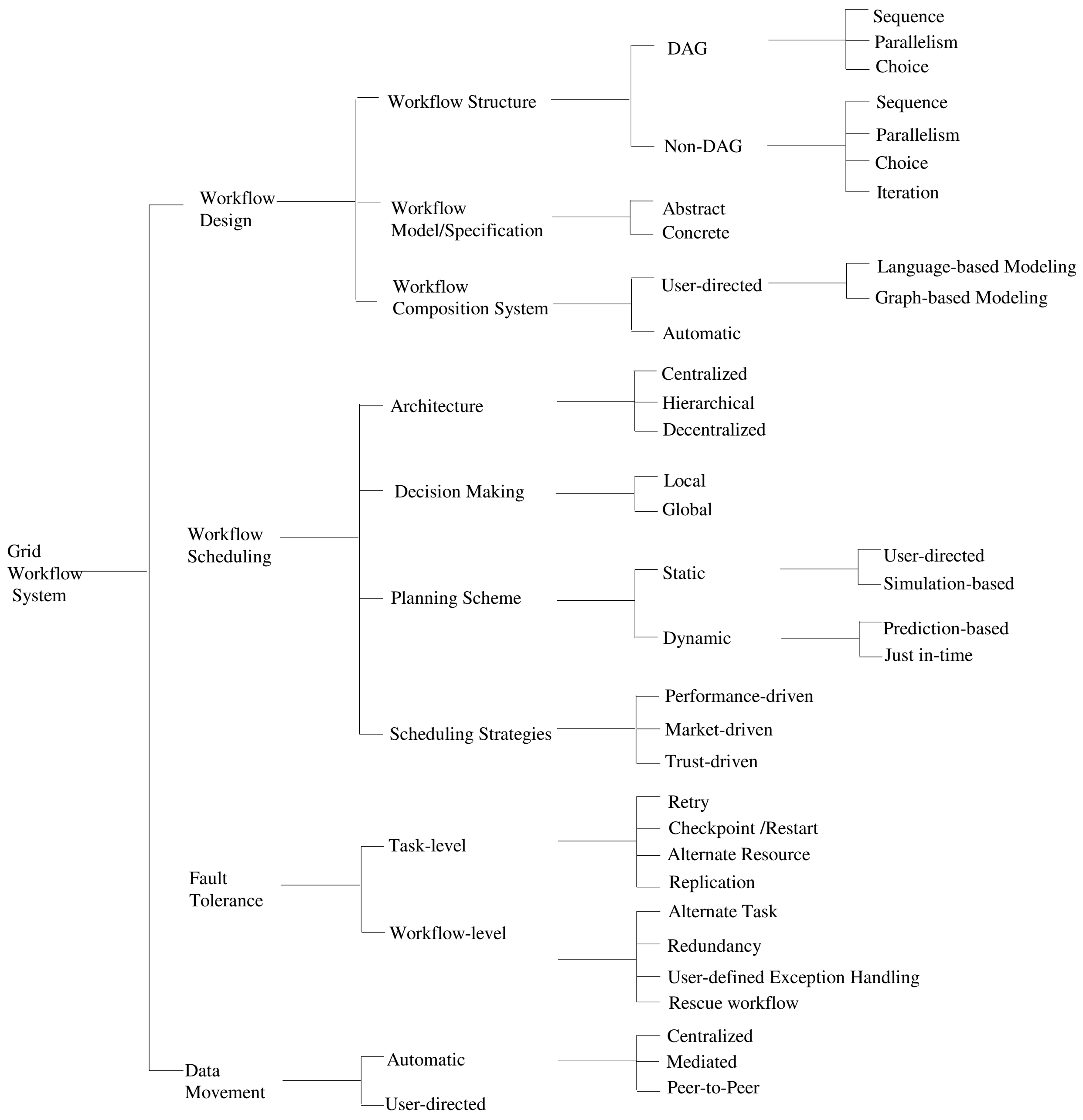

Figure 1. A taxonomy of scientific workflow systems for Grid computing.

2.1.3 Workflow Composition System

Workflow composition systems are designed for enabling users to assemble components into workflows. They need to provide a high level view for the construction of Grid workflow applications and hide the complexity of underlying Grid systems. The composition systems are categorized as *user-directed* and *automatic*.

User-directed systems require users to edit workflows directly. In general, users can use workflow languages such as Extensible Markup Language (XML) [21] for *language-based modeling* and the tools such as Kepler [2] for *graph-based modeling* to compose workflows. Compared with language-based modeling, Graphical representation is very intuitive and can be handled easily even by a non-expert user. However, workflow languages are more appropriate for storing and transfer whereas the graphical representation is required to be converted into other form for such manipulation.

Automatic systems generate workflows for users automatically according to their higher level requirements, such as data products and initial input values. Compared with user-directed systems, automatic systems are ideal for large scale workflows which are very time consuming to compose manually. However, the automatic composition of application components is challenging because it is difficult to capture the functionality of components and data types used by the components.

2.2 Workflow Scheduling

Workflow scheduling focuses on mapping and managing the execution of workflow tasks on shared resources that are not directly under the control of workflow systems.

2.2.1 Scheduling Architecture

The architecture of the scheduling infrastructure is very important for the scalability, autonomy, quality and performance of the system [11]. Three major categories of workflow scheduling architecture are *centralized*, *hierarchical* and *decentralized* scheduling schemes.

In the centralized workflow enactment environment, one central scheduler makes scheduling decisions for all tasks in the workflow. For hierarchical scheduling, there is a central manager and multiple lower-level sub-workflow schedulers. This central manager is responsible for controlling the workflow execution and assigning sub-workflows to the lower-level schedulers. In contrast with the centralized and hierarchical schemes, there are multiple schedulers without any central controller in decentralized scheduling. Every scheduler can communicate each other and schedule a sub-workflow to another scheduler with lower load.

It is believed that the centralized scheme can produce efficient schedules because it has all necessary information about all tasks in workflows. However, it is not scalable with respect to the number of tasks, the classes and number of Grid resources that are generally autonomous. The major advantage of using the hierarchical architecture is that different scheduling policies can be deployed in the central manager and lower-level schedulers [11]. However, the failure of the central manager will result in entire system failure. Decentralized scheduling is more scalable but faces more challenges to generate optimal solutions for overall workflow performance.

2.2.2 Decision Making

It is difficult to find a single best solution for mapping workflows onto resources for all workflow applications, since applications can have very different characteristics. It depends to some degree on the application models to be scheduled. In general, decisions about mapping tasks in a workflow onto resources can be based on the information of the current task or of the entire workflow. Scheduling decisions made with reference to just the task or sub-workflow at hand are called *local decisions* while scheduling decision made with reference to the whole workflow are called *global decisions* [5].

It is believed that global decision based scheduling can provide better overall results, since local decision scheduling only takes one task or sub-workflow into account. However, it also takes much more time in scheduling decision making. The overhead produced by global decision based scheduling will reduce the overall benefit and can even exceed the benefits it will produce. Therefore, the decision making of workflow scheduling should consider both overall execution time and scheduling time.

2.2.3 Planning Scheme

Schemes for translating abstract models to concrete models can be categorized into either *static* or *dynamic*. In a static scheme, concrete models have to be generated before the execution, according to current information about the execution environment, and the dynamically changing state of the resources is not taken into account. In contrast, a dynamic scheme uses both dynamic and static information about resources in order to make scheduling decisions at run-time.

Static schemes can be classified as *user-directed* or *simulation-based*. In user-directed scheduling, users emulate the scheduling process and make resource mapping decisions according to their knowledge, preference and/or performance criteria. In simulation-based scheduling, a 'best' schedule is achieved by simulating task execution on a given set of resources before a workflow starts execution. The simulation can be processed based on static information or the result of performance estimation.

Dynamic schemes include *prediction-based* and *just in-time* scheduling. Prediction-based dynamic scheduling uses dynamic information in conjunction with some results based on prediction. It is similar to simulation-based static scheduling, in which the scheduler is required to predict the performance of task execution on resources and generate a near optimal schedule for the task before it starts the execution. However, it changes the initial schedule dynamically during the execution. Rather than making a schedule prior to scheduling, just in-time scheduling only makes a scheduling decision at the time of task execution.

Planning ahead in Grid environments may produce a poor schedule, since it is a dynamic environment where utilization and the availability of resources varies over time and a better resource can join at any time. Moreover, it is not easy to accurately predict execution time of all application components on Grid resources. However, as the technology of advance reservation for various resources improve, it is believed that the role of static and prediction-based planning will increase [5] .

2.2.4 Scheduling Strategy

In general, scheduling workflow applications in a distributed system is an NP-complete problem [8]. Many heuristics have been developed to obtain near-optimal solutions to match users' QoS constraints such as deadline and budget.

Performance-driven strategies try to find a mapping of workflow tasks onto resources that achieves optimal execution performance such as minimum overall execution time. *Market-driven* strategies [10] employ market models to manage resource allocation for processing workflow tasks. Workflow schedulers with a market-driven strategy act as consumers buying services from resource providers and pay some form of electronic currency for executing tasks in workflows. Unlike performance-driven strategies, market-driven schedulers may choose a resource with later deadline if its usage price is cheaper.

Recently *trust-driven* scheduling approaches [18] in distributed systems are emerging. Trust-driven schedulers select resources based on their trust properties such as security policy, accumulated reputation, self-defense capability, attack history, and site vulnerability. By using trust-driven approaches, the overall reliability of workflow systems can be increased by reducing the chance of selecting malicious hosts and disreputable resources.

2.3 Fault Tolerance

In Grid environments, workflow execution failures can occur for various reasons such as network failure, overloaded resource conditions, or non-availability of required software components. Thus, Grid workflow management systems should be able to identify and handle failures and support reliable execution in the presence of concurrency and failures. Workflow failure handling techniques are classified as *task-level* and *workflow-level* [12]. Task-level techniques mask the effects of the execution failure of tasks in the workflow, while workflow-level techniques manipulate the workflow structure such as execution flow to deal with erroneous conditions.

Task-level techniques have been widely studied in parallel and distributed systems. They can be classified as *retry*, *alternate resource*, *checkpoint/restart* and *replication*. The retry technique is the simplest failure recovery technique, as it simply tries to execute the same task on the same resource after failure. The alternate resource technique submits failed task to another resource. The checkpoint/restart technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure. The replication technique runs the same task simultaneously on different Grid resources to ensure task execution provided that at least one of the replicas does not fail.

Workflow-level techniques are classified as *alternate task*, *redundancy*, *user-defined exception handling* and *rescue workflow*. The alternate task technique executes another implementation of a certain task if the previous one failed, while the redundancy technique executes multiple alternative tasks simultaneously. User-defined exception handling allows users to specify a special treatment for a certain failure of a task in the workflow. The rescue workflow technique generates a rescue workflow, which records information about failed tasks, during the first workflow execution. The rescue workflow is used for later submission.

2.4 Intermediate Data Movement

For Grid workflow applications, the input files of tasks need to be staged to a remote site before processing tasks. Similarly, output files may be required by their children tasks which are processed on other resources. Therefore, intermediate data has to be staged out to corresponding Grid sites. Some systems require users to manage intermediate data transfer in the workflow specification (*user-directed* approach), while some systems provide *automatic* mechanisms to transfer intermediate data.

We classify approaches of automatic intermediate data movement as *centralized*, *mediated* and *peer-to-peer*. A centralized approach transfers intermediate data between resources via a central point. In a mediated approach rather than using a central point, the locations of the intermediate data are managed by a distributed data management system. A peer-to-peer approach transfers data between processing resources.

In general, centralized approaches are easily implemented and suit workflow applications in which large-scale data flow is not required. Mediated approaches are more scalable and suitable for applications which need to keep intermediate data for later use. Since data is transmitted from a source resource to a destination resource directly, without involving any third-party service, peer-to-peer approaches save transmission time significantly and reduce the bottleneck caused by the centralized and mediated approaches. Thus, the peer-to-peer approach is more suitable for large-scale intermediate data transfer. However, there are more difficulties in deployment because it requires a Grid site to be capable of providing both data management and movement service.

3. GRID WORKFLOW SYSTEM SURVEY

A mapping of taxonomy to several existing Grid workflow systems is shown in Table 1. The detailed discussion on these systems along with identification of areas that need further work can be found in [23].

Table 1. Taxonomy mapping to Grid workflow systems.

Project	Workflow Design			Workflow Scheduling				Fault-tolerance	Data Movement
	Structure	Model	Composition System	Architecture	Decision Making	Planning Scheme	Strategies		
DAGMan [19]	DAG	Abstract	User-directed -Language-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Migration -Retrying Workflow Level -Rescue workflow	User-directed
Pegasus [6]	DAG	Abstract	User-directed -Language-based Automatic	Centralized	Local Global	Static -user-directed Dynamic -Just in-time	Performance-driven	Based on DAGMan	Mediated
Triana [20]	Non-DAG	Abstract	User-directed -Graph-based	Decentralized	Local	Dynamic -Just in-time	Performance -driven	Based on GAT manger	Peer-to-Peer
ICENI [16]	Non-DAG	Abstract	User-directed -Language-based -Graph-based	Centralized	Global	Dynamic -Prediction-based	Performance-driven Market-driven	Based on ICENI middleware	Mediated
Taverna [17]	DAG	Abstract Concrete	User-directed -Language-based -Graph-based	Centralized	Local	Dynamic -Just in-time	Performance-driven	Task Level -Retry -Alternate Resource	Centralized
GrADS [3]	DAG	Abstract	User-directed -Language-based	Centralized	Local Global	Dynamic -Prediction-based	Performance-driven	Task Level in rescheduling work in GrADS, but not in workflows.	Peer-to-Peer
GridFlow [4]	DAG	Abstract	User-directed -Graph-based -Language-based	Hierarchical	Local	Static -Simulation-based	Performance-driven	Task Level -Alternate resource	Peer-to-Peer
UNICORE [1]	Non-DAG	Concrete	User-directed -Graph-based	Centralized	User-defined*	Static -User-directed	User-defined*	Based on UNICORE middleware	Mediated
Gridbus workflow [22]	DAG	Abstract Concrete	User-directed -Language-based	Hierarchical	Local	Static -User-directed Dynamic -Just in-time	Market-driven	Task Level -Alternate resource	Centralized Peer-to-Peer
Askalon [7]	Non-DAG	Abstract	User-directed -Graph-based -Language-based	Decentralized	Global	Dynamic -Just in-time -Prediction-based	Performance-driven Market-driven	Task Level -Retry -Alternate resource Workflow level -Rescue workflow	Centralized User-directed
Karajan [13]	Non-DAG	Abstract	User-directed -Graph-based -Language-based	Centralized	User-defined*			Task Level -Retry -Alternate resource -checkpoint/restart Workflow Level -User-defined exception handling	User-directed
Kepler [14]	Non-DAG	Abstract Concrete	User-directed -Graph-based	Centralized	User-defined*			Task Level - Alternate resource Workflow Level - User-defined exception handling - Workflow rescue	Centralized Mediated Peer-to-Peer

* **user-defined** – the architecture of the system has been explicitly designed for user extension.

4. CONCLUSION

We have presented a taxonomy for Grid workflow systems. The taxonomy focuses on workflow design, workflow scheduling, fault management and data movement. We also survey some workflow management systems for Grid computing and classify them into different categories using the taxonomy. This paper thus helps to understand key workflow management approaches and identify possible future enhancements.

5. ACKNOWLEDGMENTS

We would like to acknowledge all developers of the workflow management systems described in the paper. We thank Chee Shin Yeo, Hussein Gibbins, Anthony Sulistio, Srikumar Venugopal, Tianchi Ma, Sushant Goel, and Baden Hughes (Melbourne University, Australia), Rob Gray (Monash University, Australia), Wolfram Schiffmann (FernUniversitaet in Hagen, Germany), Ivona Brandic (University of Vienna, Austria), Soonwook Hwang (National Institute of Informatics, Japan), Ewa Deelman (University of Southern California, USA), Chris Mattmann (NASA Jet Propulsion Laboratory, USA), Henan Zhao (University of Manchester, UK), Bertram Ludaescher (University of California, Davis), Thomas Fahringer (University of Innsbruck, Austria), Gregor von Laszewski (Argonne National Laboratory, USA), Ken Kennedy, Anirban Mandal, and Chuck Koelbel (Rice University, USA) for their comments on this paper. We thank anonymous reviewers for their constructive comments. This work is partially supported through the Australian Research Council (ARC) Discovery Project grant and Storage Technology Corporation sponsorship of Grid Fellowship.

6. REFERENCES

[1] J. Almond and D. Snelling. UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web. *White Paper*, October 1998,

[2] I. Altintas et al. A Framework for the Design and Reuse of Grid Workflows, *International Workshop on Scientific Applications on Grid Computing (SAG'04)*, LNCS 3458, Springer, 2005

[3] F. Berman et al. The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications(JHPCA)*, 15(4):327-344, SAGE Publications Inc., London, UK, Winter 2001.

[4] J. Cao et al. GridFlow:Workflow Management for Grid Computing. In *3rd International Symposium on Cluster Computing and the Grid (CCGrid)*, Tokyo, Japan, IEEE CS Press, Los Alamitos, CA, USA, May 12-15, 2003.

[5] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. Workflow Management in GriPhyN. *The Grid Resource Management*, Kluwer, Netherlands, 2003.

[6] E. Deelman et al. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1:25-39, Kluwer Academic Publishers, Netherlands, 2003.

[7] T. Fahringer et al. Truong. ASKALON: a tool set for cluster and Grid computing. *Concurrency and Computation: Practice and Experience*, 17:143-169, Wiley InterScience, 2005.

[8] D. Fernández-Baca. Allocating Modules to Processors in a Distributed System. *IEEE Transactions on Software Engineering*, 15(11): 1427-1436, November 1989.

[9] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[10] A. Geppert, M. Kradolfer, and D. Tombros. Market-based Workflow Management. *International Journal of Cooperative Information Systems,* World Scientific Publishing Co., NJ, USA, 1998.

[11] V. Hamscher et al. Evaluation of Job-Scheduling Strategies for Grid Computing. In *1st IEEE/ACM International Workshop on Grid Computing (Grid 2000),* Springer-Verlag, Heidelberg, Germany, 2000; 191-202.

[12] S. Hwang and C. Kesselman. Grid Workflow: A Flexible Failure Handling Framework for the Grid. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, Seattle, Washington, USA, IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.

[13] G. von Laszewski. Java CoG Kit Workflow Concepts for Scientific Experiments. *Technical Report*, Argonne National Laboratory, Argonne, IL, USA, 2005.

[14] B. Ludäscher et al. Scientific Workflow Management and the KEPLER System. *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, to appear, 2005

[15] A. Mayer et al. Workflow Expression: Comparison of Spatial and Temporal Approaches. In *Workflow in Grid Systems Workshop*, GGF-10, Berlin, March 9, 2004.

[16] S. McGough et al. Workflow Enactment in ICENI. In *UK e-Science All Hands Meeting*, Nottingham, UK, IOP Publishing Ltd, Bristol, UK, Sep. 2004; 894-900.

[17] T. Oinn et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045-3054, Oxford University Press, London, UK, 2004.

[18] S. S. Song, Y. K. Kwok, and K. Hwang. Trusted Job Scheduling in Open computational Grids: Security-Driven heuristics and A Fast Genetic Algorithm. In *19th IEEE International Parallel & Distributed Processing Symposium (IPDPS-2005)*, Denver, CO, USA., IEEE CS Press, Los Alamitos, CA, USA., April 4-8, 2005.

[19] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor - A Distributed Job Scheduler. *Beowulf Cluster Computing with Linux*, The MIT Press, MA, USA, 2002.

[20] I. Taylor, M. Shields, and I. Wang. Resource Management of Triana P2P Services. *Grid Resource Management*, Kluwer, Netherlands, June 2003.

[21] W3C. Extensible Markup Language (XML) 1.0

[22] J. Yu and R. Buyya. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In *5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)*, Pittsburgh, USA, IEEE CS Press, Los Alamitos, CA, USA, Nov. 8, 2004.

[23] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *Technical Report*, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005.