

# CROWN FlowEngine: A GPEL-Based Grid Workflow Engine

Jin Zeng, Zongxia Du, Chunming Hu, and Jinpeng Huai

School of Computer Science and Engineering, Beihang University, Beijing, China  
National Key Laboratory of Software Development Environment, Beijing, China  
Trustworthy Internet Computing Lab, Beihang University, Beijing, China  
{zengjin, duzx, hucm}@act.buaa.edu.cn,  
huaijp@buaa.edu.cn

**Abstract.** Currently some complex grid applications developing often need orchestrate multiple diverse grid services into a workflow of tasks that can submit for executing on the grid environment. In this paper, we present CROWN FlowEngine—a GPEL-based grid workflow engine for executing grid workflow instances. Besides basic functions of a conventional BPEL4WS-based workflow engine, CROWN FlowEngine has many features including hierarchical processing mechanism, multiple types of task scheduling, transaction processing, etc, which are of paramount importance to supporting workflow instances using GPEL language. CROWN FlowEngine will be adopted and widely deployed in CROWN Grid environment to support a wide range of service grid applications integration. We conduct several experiments to evaluate the performance of CROWN FlowEngine, and the results of comparing our work with GWES are presented as well.

## 1 Introduction

As the progress of grid[1] technology and the increasing prevalence of grid applications, a large numbers of resources could be accessed by drawing on grid middleware. In particular, with the appearance of OGSA (Open Grid Service Architecture)[2] architecture, some web service technologies have been successfully incorporated into grid computing to deal with resource heterogeneity and other important issues. And the resultant service grid is generally regarded as the future of grid computing. Many specifications, including OGSF, WSRF, WSDL, SOAP, etc, are introduced to standardize service grid technology. In service grid, various resources such as computers, storage, software, and data are encapsulated as services (e.g. WSRF services). As a result, resources can be accessed through standard interfaces and protocols, which effectively mask the heterogeneity.

Our CROWN (China Research and Development Over Wide-area Network)[3, 4] project aims to support large-scale resource sharing and coordinated problem solving by using service grid and other distributed computing technologies. In CROWN development, we find currently many complex grid applications developing often need integrate multiple diverse grid services into a new application. Traditional workflow technology gives this requirement a well solution. Like workflow model,

first we need to use a language to orchestrate logic and sequence relation of available grid services and create a grid workflow description document; second the grid workflow description document is deployed a execution engine, and then is scheduled and run according to defined logic and sequence relation.

In this paper, we present CROWN FlowEngine—a GPEL[5]-based grid workflow engine for executing grid workflow instances. Besides basic functions of a BPEL4WS-based workflow engine, CROWN FlowEngine has many features including hierarchical processing mechanism, multiple types of task scheduling, transaction processing, etc, which are of paramount importance to support workflow instances using GPEL language. GPEL is a grid process execution language based on the web service composition language BPEL4WS[6], and a detailed introduction about GPEL were discussed in the reference paper. CROWN FlowEngine will be adopted and widely deployed in CROWN Grid environment to support a wide range of service grid applications integration.

The rest of the paper is organized as follows. In section 2, we analyze related works in the area of grid workflow engine. The system architecture of CROWN FlowEngine is presented in section 3. In section 4, we show the implementation details and design issues. Results of performance evaluation are discussed in section 5. And in section 6, we conclude this paper and pave the way of future works.

## 2 Related Works

Grid computing offers tremendous benefits in the domains of different industry and science especially Life Sciences, Finance, Energy, Biology etc. As the appearance of service grid, we use various resources expediently through grid middleware. But while encountering a complex computing paradigm, we have to integrate and orchestrate multiple single computing units to a new application for the original requirement, so from grid service to grid workflow is a natural evolvement process in grid computing area.

Now there are some existing researches about grid workflow engine in academy domain. Jia Yu and Rajkumar Buyya of University of Melbourne publish a technical report[7] about workflow management systems for grid computing and survey current main grid workflow systems. Condor DAGMan[8] is a service for executing multiple jobs with dependencies in a declarative form. It uses DAG (Directed Acyclic Graph) as the data structure to represent job dependencies. Each job is a node in the graph and the edges identify their dependencies. Each node can have any number of “parent” or “children” nodes. Kepler[9] is a popular scientific workflow system, with advanced features for composing scientific applications, it has been extended to support web service. Taverna[10] is used to assist scientists with the development and execution of bioinformatics workflows on the Grid. FreeFluo is also integrated into Taverna as a workflow enactment engine to transfer intermediate data and invoke services. The Grid Workflow Execution Service(GWES)[11, 12] is the workflow enactment engine, which coordinates the composition and execution process of grid workflows by GWorkflowDL language. GWES supports pure web services and Globus Toolkit 4 (GT4)[13], and it is easily extendible for further execution platforms. In the section 5 we compare the performance of GWES with CROWN FlowEngine.

### 3 System Architecture

CROWN FlowEngine is a GPEL-based grid workflow engine, and Figure 1 shows its system architecture. Traditional BPEL4WS-based workflow engine provides a basic implementation of a workflow engine; however, these basic functions are not enough to satisfy the requirements for workflow engine in real grid environments.

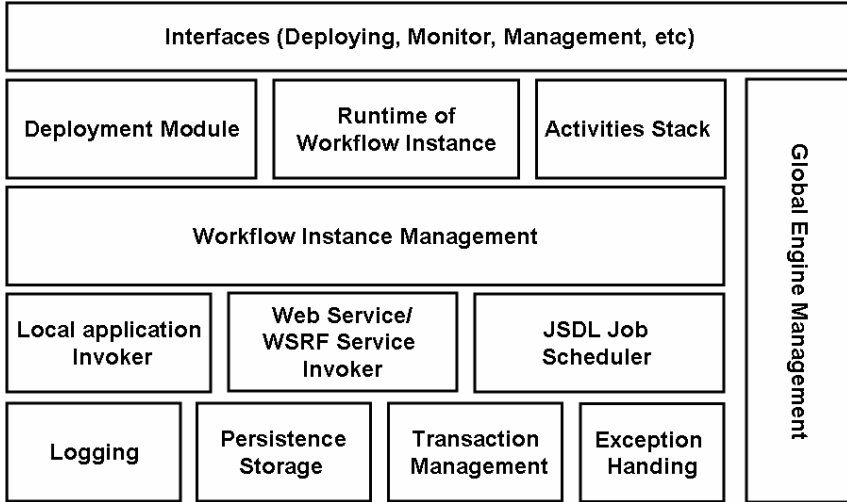


Fig. 1. System Architecture of CROWN FlowEngine

The Grid Workflow Forum[14] thinks grid workflows differ from "normal" workflows, such as business workflows or IT workflows, mainly in the following sense: stateful/stateless, reliability and performance. So when designing and developing CROWN FlowEngine, we fully consider specialties of grid environment, such state of service, reliability, performance, scheduling, etc.

First, our CROWN FlowEngine is based on GPEL, the extension of an international factual specification – BPEL4WS version 1.1, so some general workflow model created using BPEL4WS can be explained and executed. We provide stack-based explaining and executing algorithm to explaining GPEL documents and adopt multithread mechanism to executing each process instance for reducing complexity of the CROWN FlowEngine. Especially, in order to solve the problem that process instance waits longer for system resource consumption in asynchronous invoking, the method to persistencing process instance is adopted. Second, an advanced hierarchical processing mechanism enhances concurrent performance of CROWN FlowEngine. And a task processing adaptor supports multiple invocation types, including local application, general stateless web service, stateful WSRF service, and dynamic grid service schedule based on JSDL[15] for various requirements of users. Third, in order to ensure consistency and reliability, besides usual exception handling CROWN FlowEngine implements WS-Reliability[16] protocol for reliability in message layer.

It is significant to adopt a novel distributed transaction processing technology which ensures reliability and consistency of a group of key grid services. In next section we will describe these implementation details of various components and mechanisms.

## 4 Implementation Experience

### 4.1 Hierarchical Processing Mechanism

To increase quantity of the concurrent process instances in CROWN FlowEngine, we present a hierarchical processing mechanism which consists of transport layer, message layer, public service layer, workflow processing layer and corresponding interfaces[17], and Figure 2 shows its hierarchical architecture.

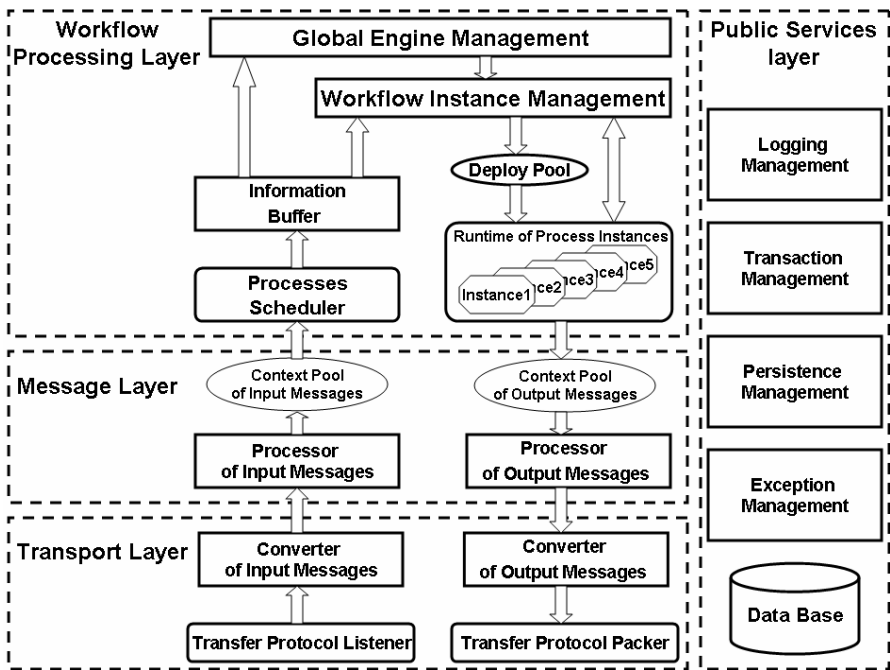


Fig. 2. Hierarchical Processing Mechanism

Transport layer is a foundation layer that can receive and transmit SOAP message bound on various transfer protocols (such as HTTP, SMTP). Therefore, this layer needs to deal with various types of user request and SOAP message transferred on various protocols. With processing content mentioned above, we design converters of in-out messages, protocol listener and packer.

Message layer ensures reliability of transferring message at invoking external application or service. Message layer provides support to WS-Reliability protocol and at least can realize one of following two requirements: first, when fault and exception

occur occasionally but not crash, all messages from sender should be delivered to receiver; second, if fault and exception keeps continuously or crash occurs, after fault and exception is eliminated or crash is resumed, all previously sent message should be delivered to receiver. In addition, in order to solve a number of concurrent requests, context pool is set in this layer.

Workflow processing layer is core of CROWN FlowEngine. Its function is to create, explain and execute a process instance, control the lifecycle of this process instance and so on. It consists of following parts:

- Global engine manager: this is the global control component; it is in charge of the management of all external messages and internal information.
- Workflow instance manager: it manages lifecycle of all process instances, including operations of creating, pausing, persistencing and resuming. According to the name and method of request message a new process instance is created.
- Runtime of process instance: it explains and executes activity in GPEL language and processes the activity of a process instance by stack-based explaining and executing algorithm.
- Processes scheduler: it adopts the policy of “First Come First Served” to deal with requests of users or external applications.

Public services layer provides the necessary functions in runtime of CROWN FlowEngine, which includes logging management, transaction management, persistence management, exceptions management, database connectivity and so on.

### 4.2 Task Scheduling

Conventional workflow engine and workflow engine based on BPEL4WS invokes tasks in a process instance is static, that is it has been decided in orchestrating. But compared with other distributed computing the characteristic of grid computing is

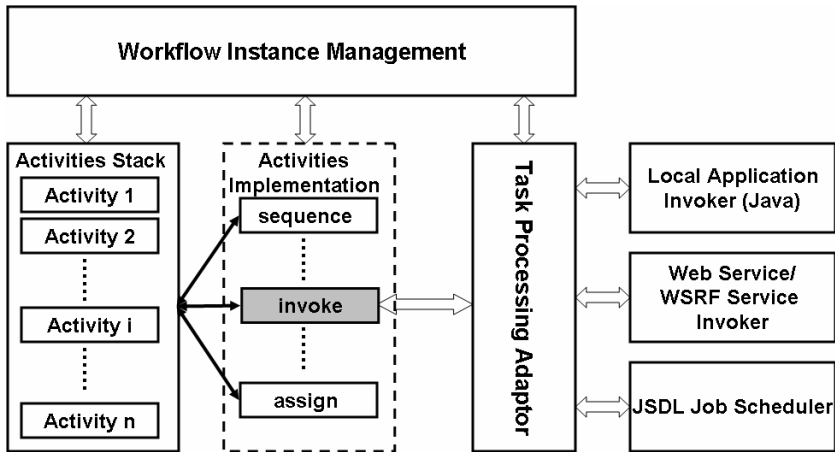


Fig. 3. Multiple Types of Task Scheduling

able to carry out job dynamic scheduling, that is a task can be located to the proper computing node according to user requirement and environment constraint. Shown as figure 3, our CROWN FlowEngine not only support local application and web service/ WSRF service call but also support grid job dynamic scheduling based on JSDL language.

In practical development, we add mostly inner property of cooperative partner description activity (partnerLink) and service invoking activity (invoke). When creating grid workflow model, user may assigns different type for certain external invoking task. In CROWN FlowEngine, workflow instance is explained and executed in activities stack. When external invoking activity (invoke) is executed, task processing adapter selects different call modes by the description at user modeling.

- Local Application Invoker: it may invoke local grid application and support currently Java language programs. This mode is mainly aimed at grid workflow developers, who need not to encapsulate the developed local application into a service in some situations. The local program is more efficient than other.
- Web Service/WSRF Service Invoker: this is a remote method invocation based on SOAP RPC. Invoking web service is basic function provided by CROWN FlowEngine. However with service grid presented and developed stateful service plays more and more important role in grid computing environment. Our GPEL support invoking standardized WSRF service. By being tested, CROWN FlowEngine is completely compatible with Globus Toolkit 4 (GT4).
- JSDL Job Scheduler: it is a remote method invoking based on SOAP Document. JSDL is used to describe the requirements of computational jobs for submission to resources, particularly in Grid environments. Noticeably, CROWN FlowEngine is a light and stand-alone grid workflow engine. CROWN FlowEngine itself has no grid job scheduling function and must together work with a external scheduler (such as CROWN Scheduler) supporting JSDL. At processing a JSDL description task, our JSDL Job Scheduler encapsulates JSDL document into SOAP message and commits it to a real external scheduler that can process JSDL document. Last, JSDL Job Scheduler parses and processes.

### 4.3 Transaction Processing Model

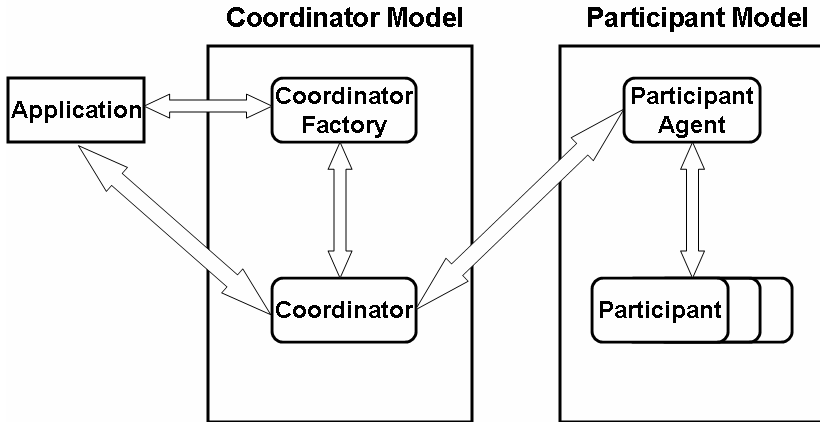
Transaction processing is an advanced characteristic of the CROWN FlowEngine, which ensure reliability and consistency of a group of key grid services.

Transaction technology is a processing mechanism that constructs a reliable application and ensures identity of all participants' output result in application. The concept of transaction originates from database research. Along with research and development of the distributed technology, distributed transaction across different resources (such as database, message middleware) is presented. Distributed transaction mentioned above can effectively apply in the closely coupled organization, but they are unable to satisfy participant's requirement for local autonomy and long transaction under the relaxed coupled grid environment. Therefore, under CROWN FlowEngine we need a new transaction technology and

model – service grid transaction to coordinate a group of key grid services in the workflow instance and ensure reliability and consistency of their result.

CROWN FlowEngine adopts BTP[18] protocol as specification of our transaction processing model. BTP is first XML-based B2B transaction processing protocol that defines the logic sequence and message types of the transaction processing. Purpose of BTP is to permit many participants owned and controlled by heterogeneous organization to coordinate running. BTP uses Two-Phase-Commit in order to ensure that the entire application program obtains the consistent result.

Figure 4 shows our transaction processing model:



**Fig. 4.** Transaction Processing Model in CROWN FlowEngine

In this model a grid workflow instance is regarded as the "external" Application. When it need to processes a group of transactional grid services the grid workflow instance sends message to Coordinator Factory. After the coordinator factory initializes an instance of Coordinator, the instance sends a request to the Participant Agent to inform it. The Participant Agent searches this node for the Participant and registers it. The instance of Coordinator commits request message to the Participant Agent and the Participant by Two-Phase-Commit and finally returns transaction result to the Application.

The Coordinator is kernel component created by the Coordinator Factory at generation of the transaction. The Coordinator coordinates the transaction processing throughout all time. During the transaction processing the Coordinator communicates with the Participant Agent time after time. One transaction processing corresponds to one coordinator and is identified with a unique ID.

The Participant Agent is essential component in transaction node. All Participants in distributed node communicate with a Coordinator via the Participant Agent. Actually in course of communication none of Participants communicate with external. The advantage of our implementation is: the Participant Agent simplifies a Participant's work; a lot of messages are centralized by a Participant Agent to control

transactions in the node. With Participant Agent, logging and committing Two-Phase-Commit become more convenient. And Participant's task is finishing business operation, but not regarding transaction implementing.

### 5 Performance Evaluation

In this section, we present the performance evaluation of CROWN FlowEngine. We will show the performance comparison of CROWN FlowEngine and GWES implementation.

The experiments are conducted on PC1 and PC2. Between the two PCs, there is an Ethernet connection of 100Mbps. PC1 has AMD Sempron 1.6 GHz CPUs, 1 GB DDR Memory and Windows XP OS and PC2 has Pentium D 3.4 GHz CPUs, 1 GB DDR Memory and Windows XP OS.

PC1 serves as the both grid workflow engines (CROWN FlowEngine and GWES) and the most simple workflow instance “Echo” (GPEL and GWorkflowDL) respectively is deployed. The workflow instance is a sequence process which can invoke an external service “HelloWorld”. There is a stateless “HelloWorld” service in the grid service container (CROWN Node Server), that is deployed on PC2. The client that sends concurrent requests to the workflow engines (PC1) using Java thread technology. We use the following metrics to do the performance evaluation. (1) Throughput. This metric is used to evaluate how many requests can be processed in one second. (2) Average response time. This metric reflects the processing efficiency under different scenarios. (3) Invocation processing time. This metric reflects the processing efficiency of external invocation. (4) Overhead of transaction processing. This metric is used to evaluate the system overhead of transaction processing under CROWN FlowEngine.

In our experiment, we hope to evaluate the metrics mentioned above of CROWN FlowEngine and GWES under different numbers of concurrent requests. All the experiments are repeated 10 times, and we report the average results.

The results of our experiment are presented in Figure 5~8. We vary the number of concurrent requests from 0 to 200 (10, 20, 30 ... 130, 140, 150, 175 and 200). Figure 6 plots the throughput of CROWN FlowEngine and GWES, while Figure 7 shows the

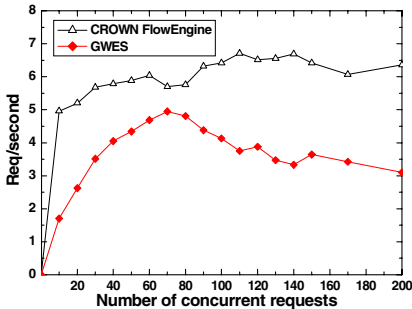


Fig. 5. Throughput vs. Num of concurrent requests (Echo)

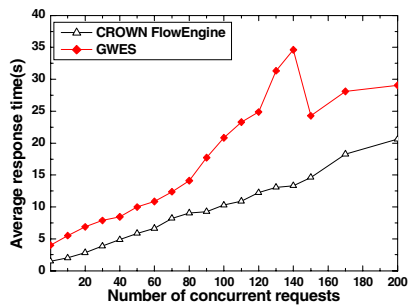
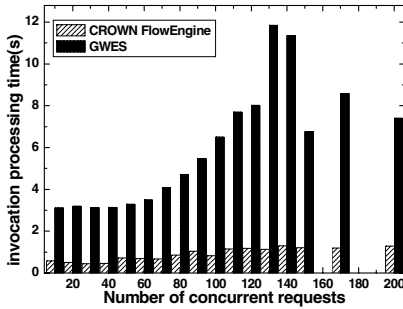
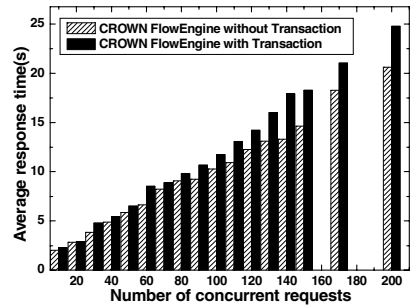


Fig. 6. Average response time vs. Num of concurrent requests (Echo)



**Fig. 7.** Comparison of Invocation processing time (Echo)



**Fig. 8.** Performance of transaction processing (Echo)

average response time for processing a request. We can see that CROWN FlowEngine outperforms GWES almost in all cases. We figure out there can be two reasons to explain the results. First, GWES is a service self and depends on third-part Application Server that deals with messages processing, in this experiment GWES is deployed under Tomcat; however, CROWN FlowEngine is a stand-alone server. Second, we design the hierarchical processing mechanism in CROWN FlowEngine to implement some buffers and pools for concurrent requests. In a general way external invocation can consume much time of a process instance processing. Therefore, on grid workflow engine side, we measure the processing time of invocation function, and the result is shown in Figure 8. The difference of invocation processing time further verifies that CROWN FlowEngine performs better than GWES. The result of Figure 9 shows that the transaction processing does not incur too much overhead to CROWN FlowEngine itself.

## 6 Conclusion and Future Works

In this paper, we present the design and implementation experience of a novel GPEL-based grid workflow engine, CROWN FlowEngine.

In order to solve the difficult problems such as stateful/stateless, reliability and performance in grid workflow, we present many feasible mechanisms and approaches including hierarchical processing mechanism, multiple types of task scheduling, transaction processing, etc. CROWN FlowEngine will include in CROWN 3.0 release that have been tested in CROWN testbed and widely use in real applications as well. We perform extensive experiments to evaluate the performance of our implementation and GWES. The results show that CROWN FlowEngine performs better than GWES under various numbers of concurrent requests.

We will perform more tests on all components of CROWN FlowEngine to make it stable to use. Also we will further design and develop a console for CROWN FlowEngine to monitor, manage and analyze process instances. In addition, we hope to perform further study on performance issues.

## Acknowledgement

We thank our colleagues; Bing Bu and Dou Sun implement much programming work, and Jiabin Geng, for his help with client-side programming work and setting up the experimental environment.

This work is part of the results of the project CROWN and is supported by the NSF of China under Grant 90412011 and by China National Natural Science Funds for Distinguished Young Scholar under Grant 60525209, partly by National Basic Research Program of China 973 under grant no. 2005CB321803, and partly by Development Program of China 863 under grant no. 2006AA01A106.

## References

1. Foster, I., Kesselman, C.: *The Grid: Blueprint for a New Computing Infrastructure*, 2nd edn. Morgan Kaufmann, San Francisco (2004)
2. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: Grid Services for Distributed System Integration. *IEEE Computer* 35, 37–46 (2002)
3. CROWN project: <http://www.crown.org.cn/en>
4. Huai, J., Hu, C., Li, J., Sun, H., Wo, T.: CROWN: A Service Grid Middleware with Trust Management Mechanism. *Science in China Series F* 49, 731–758 (2006)
5. Wang, Y., Hu, C., Huai, J.: A New Grid Workflow Description Language. *IEEE International Conference on Services Computing*. IEEE Computer Society Press, Los Alamitos (2005)
6. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S., Thatte, S.: *Business Process Execution Language for Web Services version 1.1* (2003), <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>
7. Yu, J., Buyya, R.: *A Taxonomy of Workflow Management Systems for Grid Computing*. Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne (2005)
8. Thain, D., Tannenbaum, T., Livny, M.: *Condor and the Grid. Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, NJ, USA (2003)
9. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: *Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows* (2005)
10. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. *BIOINFORMATICS*, vol. 20 (2004)
11. Neubauer, F., Hoheisel, A., Geiler, J.: Workflow-based Grid applications. *Future Generation Computer Systems* 22, 6–15 (2006)
12. The Grid Workflow Execution Service: <http://www.gridworkflow.org/kwfggrid/gwes/docs/>
13. Globus Toolkit: <http://www.globus.org/toolkit/>
14. The Grid Workflow Forum: <http://www.gridworkflow.org/>
15. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: *Job Submission Description Language (JSDL)* (2005), <http://forge.gridforum.org/projects/jsdl-wg>
16. Iwasa, K., Durand, J., Rutt, T., Peel, M., Kunisetty, S., Bunting, D.: *OASIS Web Services Reliability* (2004), <http://docs.oasis-open.org/wsrn/ws-reliability/v1.1/wsrn-ws-reliability-1.1-spec-os.pdf>

17. Bu, B.: Research and Implementation of a Lightweight Workflow Engine based on Web Service, Master Thesis, in School of Computer Science and Engineering, Beihang University (2006)
18. Ceponkus, A., Dalal, S., Fletcher, T., Furniss, P., Green, A., Pope, B.: OASIS Business Transactions Technical Committee. Business Transaction Protocol?BTP?Version 1.0 (2002), [http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP\\_cttee\\_spec\\_1.0.pdf](http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf)