# Architectural Plan for Constructing Fault Tolerable Workflow Engines Based on Grid Service

***Saeed Parsa***
*Department of Computer Engineering*
*Iran University of Science and Technology*
*Tehran, Iran*
E-mail: parsa@iust.ac.ir

***Alireza Azimi***
*Department of Computer Engineering*
*Islamic Azad University, Sari Branch,*
*Sari, Iran*
E-mail: Azimi@iausari.ac.ir

**Abstract**

*In this paper the design and implementation of fault tolerable architecture for scientific workflow engines is presented. The engines are assumed to be implemented as composite web services. Current architectures for workflow engines do not make any considerations for substituting faulty web services with correct ones at run time. The difficulty is to rollback the execution state of the workflow engine to its state before the invocation of the faulty web service. To achieve this, three components for fault diagnosis, recording the execution state of the workflow and substitution of faulty web services, at run time, are considered in our proposed architecture. The applicability of the proposed architecture is practically evaluated by applying it to design of three different scientific workflow engines.*

***Keywords:*** *workflow engine, SOA, web service, grid service*

## 1. Introduction

The aim has been to design a fault tolerable architecture for implementing workflow engines to run scientific applications across grid networks. Scientific workflow engines allow scientists to specify and run large computational experiments across wide area networks. To support extensibility, maintainability and rapid modifiability, workflow engines could be preferably implemented as composite web services in which the member web services could be easily extended and replaced with new ones. Specifically, when a member web service fails to execute appropriately it is desirable to be able to replace it with another web service, at run time [1, 2, 3]. There are a few faults tolerable architectures proposed for implementing scientific workflow engines [4, 5, 6, 7]. To enhance fault tolerability, the design and implementation of a new architecture for implementing scientific workflow engines as composite substitutable web services is presented in this paper. The proposed architecture provides a component to determine appropriate web services dependent on the user requests. The user requests may include quantitative quality features of the desired services. Considering the composability, interoperability, Compatibility and substitutability of the available web services, the system automatically selects the appropriate web services to serve the user.

Composability is a key, albeit long-term, benefit of moving towards a service-oriented architecture [8]. It is an aspect of service design that can only be effectively realized. To evaluate composability of the explored web services, comprehensive descriptions of the operations and interfaces of the web services are required. Standard

descriptions of the structure, semantics and quality of web services are provided within UDDI (Universal Description Discovery and Integration) services across the internet.

Interoperability is the ability of a service to work with the other services without special effort across the internet [9]. To provide interoperability, the architecture proposed in this paper, a component, called workflow management, keeps records of composable services and their cooperation protocols.

Compatibility of web services states the fitness of services that interact with each other and closely relates to substitutability of service peers [10]. Compatibility is concerned with both the static features and dynamic behaviors of service peers. In the proposed architecture, the coordinator component describing dynamic behaviors of web services is used.

To avoid a composite web service malfunctioning due to the failure of its member web services the substitutability of the faulty members with similar ones is crucial. Substitutability of member services in a composite web service refers to the ability of using candidate services to replace faulty services, dynamically. Substitutability improves robustness and independency of composite web services from their member services. The proposed architecture assigns a distinct component to dynamically detect and substitute faulty member services in a composite web service, at run time.

The remaining parts of this article are organized as follows: The related works are described in Section II. Section III presents a new fault tolerable architecture for scientific workflow engines. In section IV noted points in implementing propose workflow engine will be studied. The proposed architecture is evaluated both in practice and theory in Section V. The conclusions and future works are presented in Section VI.

## 2. Related Work

In this section the difficulties and problems concerning current trends and tools in design and development of workflow management systems are described. A major task of a workflow management system is to determine the most appropriate location for executing jobs assigned to the workflow manager. However it is observed that even GridAnt [11] and Triana [12, 13] which are two well known workflow management systems ask the user to define the exact location of the jobs. Askalon [14] and Condor-G [15] determine the most appropriate location for executing jobs by referring to the information which is frequently collected about interconnections and volume of data to be transferred between the jobs and their activities. However, to determine the exact location of activities at run time semantics descriptions of the activities are required. Semantics description of the activities implemented as composite web services could be simply attained via the UDDI descriptions. These descriptions could be applied to locate composite web services.

In our proposed architecture, descriptions of each composite web service, including a unique identifier, sequence of participating web services, the interconnections with other composite web services, are dynamically recorded and kept in a common data structure. The information collected in this data structure could be further applied to locate faulty web services at runtime. Runtime errors are detected by a component, Fault-Manager, in our proposed architecture. The Fault-Manager component applies exception handling mechanism to detect predicted error prone conditions. Predefined abnormal behaviors and expected execution time could be applied as a basis for detecting faulty web services. Considering the execution log of a faulty web service it is

possible to rollback the program execution to its state before the execution of the faulty web service. The faulty web service could be replaced with a correct one.

Fault detection and recovery has been only applied at hardware an operating system levels [11]. Workflow management systems such as Condor-G [15, 16], Taverna [8] and GridAnt [11, 17], UNICORE [18] could detect faults at workflow and task levels. These systems provide the users with a history of the faults detected at runtime. Automatic recovery from faults at workflow level is performed by Askalon [14], Triana [12,13], Pegasus [19] and GWEE [20] workflow management systems. A few number of known workflow management systems such as Chemomentum [21], Escogitare [22] and Triana [13] could detect runtime faults at task level. Job-related faults such as deadlock, livelock, memory leak, uncaught exceptions, missing shared libraries and incorrect output results are examples of faults at task level. There have been no provisions for automatic recovering from such faults because workflow systems have been mostly concerned with the flow of data and control between the task rather than the tasks contexts.

## 3. The Proposed Architecture

The workflow management system proposed in this section complies with a known model called workflow coalition [22, 23, 24]. The main idea behind the design of this system has been to provide an engine capable of managing workflows on heterogeneous platforms with high reliability and the possibility for automatic fault detection and recovery. The proposed architecture also supports dynamic replacement of existing workflow activities with modified versions of the activities. Implementing the workflow activities as loosely coupled web services it has been possible to dynamically replace existing web services with the ones with higher quality. To achieve this, a quality management component, QOS, is considered in the proposed architecture. The QOS component is described in section A. Dynamic replacement of the workflow activities may be also required when recovering from runtime faults. After a runtime error is detected, the faulty task could be replaced with a correct one. Fault detection is carried on by a fault monitoring component called Fault-analyzer. The Fault-analyzer component is a service provided by the Fault-Management component, described in section B. A distinct component called WorkFlowDef provides the users with an environment to define and modify workflows. The WorkFlowDef component is described in section C. User requests are handled by a component called Service-enquires. The Service-enquires component is described in section D. The appropriate services to handle the user's enquiries are selected by the SearchAndrecovery component, described in section E. New instances of the selected services are created by the WS-Instance component described in section F. All the data and the messages to be passed between the workflow activities are kept by the ComponentLib component are described in section G. A log of activities participating in each job executing by the workflow engine is kept by the Context-Management component of the workflow management service described in section H.

The overall architecture of the proposed workflow management system is presented in Figure 1. The architecture components are further described in the following sub sections.

### *A. The QOS Component*

The QOS component provides an interface to search the internet for the web services similar to the ones applied in the workflow system. The component uses the UDDI registries across the internet to find qualified services considering the services qualities defined by the users. After the user approval, the qualified component is replaced with its corresponding component in a library called ComponentLib. The component is also replaced with it corresponding component within the workflow engine.

### *B. The Fault-Management Component*

The proposed architecture supports Fault tolerance through the Fault-management component. As shown in Figure 1 this component comprises three sub components named Fault-Controller, WS-Recovery and Fault-Analyzer.

The Fault-Analyzer component receives a notification from the workflow program exception handler whenever any predefined faulty behavior is detected. Predefined faulty behaviors are catch by the program exception handler. The Fault-Analyzer component also detects faults by controlling the execution time of the web services after a fault is detected. The Fault-Analyzer reports the detected fault to the Fault-Controller component. The Fault-Controller component after a fault is detected the WS-Recovery component replaces the detected faulty web service with correct one. Fault recovery is carried out by the WS-Recovery Component. This component uses the workflow program execution log to rollback the execution of the current job to the state immediately before the start of the faulty activity. All the information concerning the execution of the current web services are kept by the Context-Management component in a log file. The WS-Recovery component asks the SearchAndRecovery component to look for a web service similar to the faulty one. The SearchAndRecovery component gets the description of the faulty web service from a component called Fault-Controller component.

### *C. The WorkFlowDef Component*

A new workflow could be specified trough the environment provided by the WorkFlowDef component. WorkFlowDef component applies workflow definition and description tools such as workflow creator in .Net framework to define rules concerning workflows. These rules are farther applied by the workflow engine to build specific workflows in accordance with the user requirements.

### *D. Service-Enquiry*

The Service-Enquiry component receives high level descriptions of compound services from the users. The component breaks down the compound service requests into simple sub-requests. The resultant sub-requests are analyzed and used by the component to look for appropriate web services. The selected web services are further applied to build service orchestrations reply to the user requests.

In fact a workflow defines a set of activities which may be executed in any order dependent on the decisions made. Each activity within a workflow may be defined as a

sub-workflow. Therefore, a complicated user's enquiry may be defined as a workflow where each activity itself may be defined as a sub-workflow.

### *E. The SearchAndRecovery Component*

After a faulty web service is detected, the SearchAndRecovery component is invoked to look for corresponding web service across the internet. The component collects information describing the faulty web service from a table handled by the Context-Management component. The Context-Management component fills this table trough the user enquiries.

### *F. WS-Instance*

After a user enquiry is interpreted and analyzed by the Service-Enquiry component, new instance of the web services responding to the user enquiry are build by the WS-Instance component. These instances are applied by the workflow engine to run the workflow in accordance with the user enquiry. The users could follow the requests trough this instances.

### *G. The ComponentLib Component*

The ComponentLib keeps a record of each workflow and each activity or web service being executed with in the workflow engine. After a workflow is terminated all its records are wiped off. These records are applied by the context-management component to rollback the faulty component to safe states.

### *H. WorkFlow Management*

Workflow management component is a special description contained list of participating services in composition, their orders, and their connection method, coming and going services between them for making composition services. The component is a standard mechanism for services to distribute and record of content descriptions which presented for activities. It formulated from 3 parts of coordinator component, context management component and selecting protocols. Coordinator component has the duty of making relation between web services and connection with other framework components. Context management component makes a common field for partial services of a composition that contained a place for maintaining common information field between services and state maintenance of composition services.

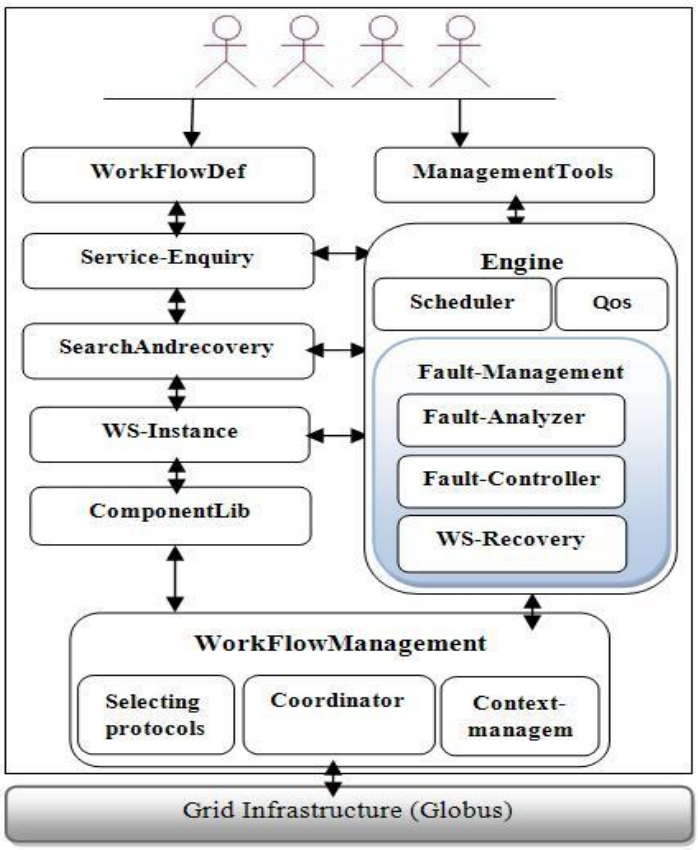


*Figure 1. Purposed architecture*

## 4. Workflow Engine Implementation

Proposed workflow engine comprises three items named scheduler, fault tolerance and quality of services. Regarding to predicate mechanism of fault tolerance, when error occurs in any task execution, workflow engine in addition to call the SearchAndRecovery component, identify machines which had requested web services. Also, after insuring from service quality measurement, the workflow engine selects suitable resource and sends a task by encountered error for executing over it.

Regarding to scheduling mechanism, workflow method is centralized. proposed scheduler after receiving workflow, identify tasks and do workflow scheduling by selecting appropriate machine for executing each tasks without informing users (in comparison with other machines which had better state). Rated measurement of resources (which introduced to workflow definition part) includes activity rate, speed and processor architecture, memory capacity and network-load and also type of operating system. Scheduler records data related each task in a log file at runtime. This information could gain suitable procedures in future runs. Although as a case item it could by automated for next decision making, now, the workflow engine hasn't predicted any planning.

Workflow engine implementation environment is product of visual studio .Net 2008 Microsoft Company, and based on software platform Microsoft .Net framework 3.5.

## 5. Evaluation

In this section, evaluation of workflow engine is presented. An architecture proposed in this paper, Condor (Condor-G) workflow engine [15] and MyGWFMiddlWare workflow engine [25] done previously at this faculty, have been selected, and sample workflow will be executed on them. Besides, in order to compare these three workflow engines accurately, a reassessment has been done on the proposed architecture based on the previous comparison evaluation.

One of the useful and appealing items of statistics is data number clustering. For example, let us cluster the Following data based on numbers 3, 10 and 14. Three machines A, B and C compute the square subtracting of input data 3, 10 and 14 successively and write in files A.Out, B.Out and C.Out. Figure 2 is shown the entrance file and the internal files of clustering.

| In.Dat | A.Out(Cluster3) | B.Out(Cluster10) | C.Out(Cluster(14) |
|---|---|---|---|
| 6 | $(6-3)^2=9$ | 16 | 64 |
| 12 | $(12-3)^2=81$ | 4 | 4 |
| 13 | $(13-3)^2=100$ | 9 | 1 |
| 4 | 1 | 36 | 100 |
| 3 | 0 | 49 | 121 |
| 9 | 36 | 1 | 25 |
| 16 | 169 | 36 | 4 |
| 0 | 9 | 100 | 196 |

*Figure 2. The entrance file and the internal files of clustering*

$$(|6-9| = 3)\ ?\ (|6-16| = 10)\ ?\ (|6-36| = 30) \Rightarrow 6 \in 3 \text{ Cluster}$$

| Out.Dat |
|---|
| 6 > 3 |
| 12 > 10 |
| 13 > 14 |
| 4 > 3 |
| 3 > 3 |
| 9 > 10 |
| 16 > 14 |
| 0 > 3 |

*Figure 3. The final result of clustering*

Machine M evaluates input file (In.Dat) and produces the clustered data files (Out.Dat) as follow. Machine M reads the first file line of In.Dat and calculates the supposed data of subtracting absolute value (no.6) C.Out, (no.16) B.Out, (no.9) A.Out. The three result numbers are compared and the lowest one indicates the belonging of no.6 to that cluster. Thus, no.6 belongs to cluster no.3. Figure 3 illustrates results of clustering.

The operating environment considered for this purpose has been chosen for 10 computers connected to the internet with different IP addresses in one local network with three Windows operating systems (MS Windows XP, MS Windows 2000 and MS Windows 2003) and Fedora Core 4, successively for the job sending to proposed workflow engine and Condor. The connective protocol of machines is TCP/IP for both

workflow engines and both of them take advantage of the Intel processors with computed speeds and Pentium 4 category.

Experimental data to assimilate clustering functions are inside the file In.Dat and are as many as 1000000 of four digital unsigned integer. Since the data clustering in a single cluster is not that meaningful, the job is commenced with two clusters and naturally two machines. The procedure of workflow continues up to 10 clusters. The average results deriving from 10 times run for each workflow (a two cluster clustering, a three cluster clustering … and up to 10 cluster clustering) on the proposed engine, is shown in figure 4.

| clustering no. | Time (ms) |
|---|---|
| 2 | 232146.000 |
| 3 | 239671.200 |
| 4 | 243105.500 |
| 5 | 246258.000 |
| 6 | 267878.500 |
| 7 | 270142.020 |
| 8 | 281101.100 |
| 9 | 300841.500 |
| 10 | 308191.433 |

*Figure 4. The result of clustering using the proposed engine*

The comparative diagram of executing clustering process over the proposed architecture and over two selected engines is shown in figure 5. Paying attention to the following points is noteworthy in analyzing the diagram.

Since Condor doesn't have the necessary mechanism to transfer the file, the internal files are transmitted to the final file after being produced by manual method; while, this process in the proposed model through connective network protocols and far from the user's eyes is carried out. Hence, the overflow resulting from this transmission must be considered in these evaluations.

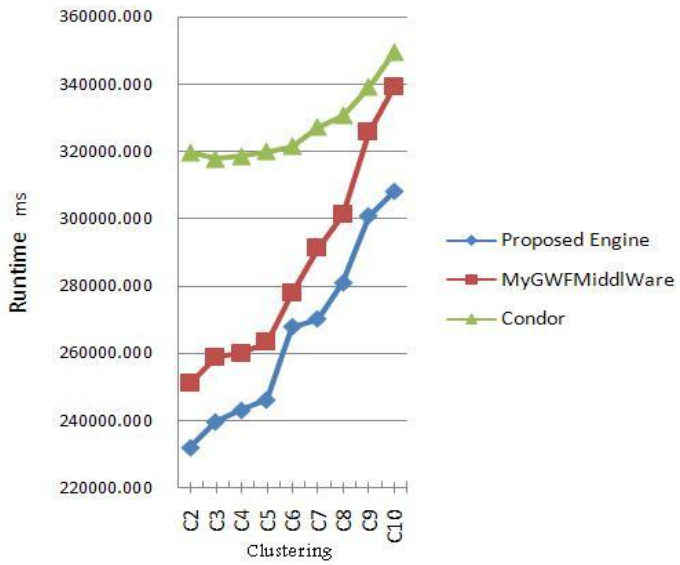


*Figure 5. The comparative diagram of executing clustering process over the proposed architecture and over two selected engines.*

As it was raised earlier 10 machines were considered to execute the workflow. Because in the first timed process of workflow the proposed model has the potentiality of choosing suitable machines for executing the workflow, the time limit for the proposed engine is far less in comparison with Condor. This condition, gradually, with the increase of clusters, and because of the busy machines has changed so that the time of executing workflow in clustering 8, 9 and 10 by far increased and has approached to the Condor time.

The reason for the appearance of this phenomenon can be due to the lack of ability of scheduler in choosing the right machines. Because with the passage of time when the number of clusters go up, the machines are devoted to the workflow; hence, the target of the scheduler for choosing the right machine becomes smaller and smaller. For example, in order to choose the ninth machine for computing the ninth cluster, only two machines, to assign the workflow, are at the disposal of the scheduler. As a result with the increase of the machines we can assign scheduler to the society of a more varied target to choose more suitable resources.

## 6. Conclusion and Future Work

The aim of this paper is to present scientific workflow management system on the basis of the web service over the grid network. Running the workflow and the inner tasks in the proposed model, through the published web services exists in the grid network machines. In this paper, the problems workflow management systems have been considered and in continuation a certain solution in the form a proposed architecture with exploiting the service-oriented architecture (SOA) was presented. Also, it was explained that how the proposed architecture is harmonized with workflow management systems. In this line, a definite infrastructure is defined for managing workflow which is in connection with Globus standards, W3c and workflow coalition. In the end a workflow execution language (WFEL) for GT3 (Globus-Toolkit 3) environment was presented. In line with the continuation the carried out activities one can identify the current impedance to activate the workflow management systems and with regard to the suitable mechanism you can reduce and finally remove them. The advent of this affair can enormously reduce the worried of the users connected to the grid. Among the other activities ahead one can point to the improvement of the proposed architecture; can better the fault tolerance in the time of executing workflow, and with the increase of the tag range of workflow execution language (WFEL) you can improve the quality of service (Qos) presented. In addition, overflow of jobs, the expenses of the exploiting the resources and the scalability of the scheduler at the runtime can be taken into consideration.

## References

[1] M. K. S. Chan, J. Bishop, J. Steyn, L. Baresi, and S. Guinea' ''A Fault Taxonomy for Web service Composition'', *In Proc of the Workshop on Engineering Service-Oriented Applications: Analysis, Design and Composition (WESOA '2007)*, Vienna, Austria, 2007.

[2] Schafer, M., Dolog, P., Nejdl, W, ''Engineering compensations in web service environment'', Fraternali, P., Baresi, L., Houben, G.-J. (Eds.). Berlin, Germany: Springer-Verlag, 2007

[3] Rubing Duan, Thomas Fahringer, Radu Prodan, Jun Qin, Alex Villaz´on and Marek Wieczorek, *Real World Workflow Applications in the Askalon Grid Environment*, P.M.A. Sloot et al. (Eds.).Berlin, Germany: Springer-Verlag 2005

[4] Anne H. H. Ngu, Shawn Bowers, Nicholas Haasch, Timothy M. McPhillips and Terence Critchlow, *Flexible Scientific Workflow Modeling Using Frames*, B. Lud¨ascher and Nikos Mamoulis (Eds). Berlin, Germany: Springer-Verlag 2008.

[5] Daniel Craw and Ilkay Altintas, *A Provenance-Based Fault Tolerance Mechanism for Scientific Workflows* In: Provenance and Annotation of Data and Processes (2008), p. 152-159.

[6] S. Hwang and C. Kesselman, ‘’Grid Workflow: A Flexible Failure Handling Framework for the Grid’’, *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, p.126, 2003.

[7] Tom Oinn , Mark Greenwood , Matthew Addis , M. Nedim Alpdemir , Justin Ferris , Kevin Glover , Carole Goble , Antoon Goderis , Duncan Hull , Darren Marvin , Peter Li , Phillip Lord , Matthew R. Pocock , Martin Senger , Robert Stevens , Anil Wipat and Chris Wroe, *Taverna: Lessons in creating a workflow environment for the life sciences*, .Concurrency and computation: Practice and experience.13 Dec 2005, John Wiley & Sons, Ltd.

[8] Thomas Erl, ‘’Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services’’, Prentice Hall, 2004.

[9] Sebastian Truptil, Frederick Benaben, Pierre Couget, Matthieu Lauras, Vincent Chapurlat, and Herve Pingaud. ‘’Interoperability of Information Systems in Crisis Management; *Crisis Modeling and Meta modeling*, London, England: Springer 2008, P 583-594.

[10] Chen Shu, Wu Guo Qing and Xiao Jing, ‘’A Process Algebra Approach for the Compatibility Analysis of Web Services’’ *2008 Second International Conference on Future Generation Communication and Networking, IEEE Computer Society Washington*, DC, USA. 13-15 December

[11] Kaizar Amin and Gregor von Laszewski, *GridAnt: A Grid Workflow System. Manual*, February 2003. [Online]. Available: http:// www.globus.org/cog/projects/gridant

[12] I. Taylor, M. Shields and I. Wang, ‘’Resource Management of Triana P2P Services’’, Grid Resource Management, Kluwer, Netherlands, p. 451-462, 2003.

[13] I. Taylor, M. Shields, I. Wang, and A. Harrison, The Triana Workflow Environment: Architecture and Applications, In I. Taylor, E. Deelman, D. Gannon, and M. Shields, editors, *Workflows for e-Science*, pages 320-339. Springer, New York, Secaucus, NJ, USA, 2007.

[14] T. Fahringer, R. Prodan, R. Duan, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, M. Wieczorek ASKALON: A Grid Application Development and Computing Environment, *6th International Workshop on Grid Computing*, (C) IEEE Computer Society Press, November 2005, Seattle, USA

[15] J. Frey, T. Tannenbaum, M. Livny and S. Tuecke, ‘’Condor-G: a computation management agent for multi-institutional grids’’, *Proceedings of the 10th IEEE Symposium on High- Performance Distributed Computing*, 2001. See also http://www.globus.org/grid_software/computation/condor-g.php

[16] D. Thain, T. Tannenbaum, and M. Livny, *Distributed Computing in Practice: The Condor Experience, Concurrency and Computation: Practice and Experience*, 2005, pp. 323-356

[17] Von Laszewski G et al. GridAnt: A client-controllable Grid workflow system. *Proc of the 37th Hawaii International Conference on System Sciences*, Hawaii, HI, 2004. IEEE Computer Society Press: Los Alamitos, CA, 2004.

[18] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder, UNICORE - From Project Results to Production Grids, L. Grandinetti (Edt.), Grid Computing: The New Frontiers of High Performance Processing, *Advances in Parallel Computing 14*, Elsevier, 2005, pages 357-376

[19] N. Mandal, E. Deelman, G. Mehta, M-H. Su, and K. Vahi, Integrating Existing Scientific Workflow Systems: The Kepler/Pegasus Example, *Proceedings of the Second Workshop on Workflows in Support of Large-Scale Science (WORKS'07), in conjunction with the IEEE International Symposium on High Performance Distributed Computing Monterrey, CA*, June 2007

[20] E. Elmroth, F. Hernandez and J. Tordsson. A light-weight Grid workflow execution service enabling client and middleware independence. *Proceedings of the Grid Applications and Middleware Workshop*, PPAM 2007, Springer-Verlag, Lecture Notes in Computer Science.

[21] B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, S. Sild, U. Maran, P. Bala, E. del Grosso, M. Casalegno, N. Piclin, M. Pintore, W. udholt and K. Baldrige, Chemomentum - UNICORE 6 based infrastructure for complex applications in science and technology, *Proceedings of 3rd UNICORE Summit 2007 in conjunction with EuroPar*, 2007, Rennes, France

[22] Workflow Management Coalition, ''Workflow Management Application Programming Interface (Interface 2&3) Specification'', WFMC-TC-1009, Version 2.0, 2003.

[23] Workflow Management Coalition, ''Workflow Process Definition Interface - XML Process Definition Language'', WFMC-TC-1025, Version 1.0, 2002.

[24] Workflow Management Coalition, ''Reference model and API specification'', WfMC-TC00-1003, 1996.

[25] H.Mahdikhani, "Design And Implementation Of Architecture of Workflow Engine With Grid Service " , *Msc. Eng. thesis, Dep. Of Computer Science and Technology. IUST, Iran*, 2002

[26] H. Shan and L. Oliker, ''Job Super scheduler Architecture and Performance in Computational Grid Environments'', *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, p. 154-163, 2003.

[27] G. Wasson, N. Beekwilder, M. Morgan and M. Humphrey, ''WS-Resource Framework on .NET'', *In 13th IEEE International Symposium on High Performance Distributed Computing (HPDC-1304)*, pp. 258-259, 2004.

[28] J. H. Abawajy, ''Fault-Tolerant Scheduling Policy for Grid Computing Systems'', *In 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, IEEE Computer Society (CS)*, pp. 238-244, 2004.

[29] Y. Jin, Z. Wu, S. Deng and Z. Yu, ''Service-Oriented Workflow Model'', *19th International Conference on Advanced Information Networking and Applications (AINA'05), IEEE, Volume 2 (INA,, USW,, WAMIS,, and IPv6 papers)*, pp. 484-488, 2005.

[30] K. Amin S. Hampton G. von Laszewski, B. Alunkal and S. Nijsure, *Gridant-client-side workflow management with ant*, whitepaper, July, 2002. 2002.

[31] J. Yu and R. Buyya, *A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing*, Volume 3, Numbers 3-4, Pages: 171-200, Springer Science+ Business Media B.V., New York, USA, Sept. 2005.

[32] D. Laforenza, R. Lombardo, M. Scarpellini, M. Serrano, F. Silvestri, P. Faccioli, Biological Experiments on the Grid: A Novel Workflow Management Platform, pp. 489-494, *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)*, 2007