QoS Aware and Service-Oriented Workflow on GRID

Li Guo, Steve McGough, Asif Akram
David Colling, Janusz Martyniak, Marko Krznaric
Imperial College London, UK
{liguo,asm,aakram,dc1,Jm,mk}@doc.ic.ac.uk

Abstract

As the main computing paradigm for resource-intensive scientific applications, Grid[1] enables resource sharing and dynamic allocation of computational resources. Large-scale grids are normally composed of huge numbers of components from different sites, which increases the requirements of workflows and quality of service (QoS) upon these workflows as many of these components have real-time requirements.

In this paper, we describe our web services based QoS-aware workflow management system(WfMS) from GridCC project[7] and show how this WfMS can aid to ensure workflows meet the pre-defined QoS requirements and optimise them accordingly.

1. Introduction

The Grid has emerged in recent years as a paradigm for allowing users to obtain access to large quantities of computing resources while providing a forum in which resource owners make their services available to a wide audience. In order to perform constructive science, a scientist will in general need to perform multiple tasks in order to achieve their goal. These may include such things as configuring an instrument, collecting and storing relevant data from the instrument, processing of this information and potentially further iterations of these tasks. This can be seen as a set of tasks which interact with each other in order to achieve the final result and can be considered as a workflows. Workflow management systems (WfMSs) have been used to support various types of e-science workflows on the grid. In addition, as the grid has evolved into a Service Orientated Architecture (SOA)[2], web services[3] emerge and are used as the de-facto communication mechanism for the distributed partners of the workflow in grid.

Large-scale grids are normally composed of huge numbers of components from different sites. The selection of the best resources to use within the Grid is complicated due to its dynamic nature with resources appearing and disappearing without notice and the load on these resources changing dramatically over time. These are issues that the scientist will, in general, not wish to be concerned about. This increases the requirements of QoS. QoS constraints for interactions between the different Grid components thus becomes crucial to enable resources oriented workflows. In such workflow processes, service providers and consumers define a binding agreement or contract between the two parties, specifying QoS properties such as response time of particular resources for certain tasks, etc. Management of such QoS directly impacts success of parties' participating in the coordination processes.

In this paper we present some ideas and results on a QoS-aware WfMS from the GRIDCC project. It shows how QoS components can be possibly used in workflows in grid environment. In Section 2 we present related work. The overall architecture of GridCC is presented and briefly explained in Section 3. In section 4 we analyse and classify QoS requirements in Grid.

The architecture our QoS based WfMS is explained in Section 5. Section 5 also contains a more detailed breakdown of our system components. Conclusion and possible future work are given in Section 6.

2. Related Work

Many Gird workflow systems and tools exist, such as: Askalon[8], DAGMan[9], Grid-Flow[10], GridbusWorkflow [11], ICENI[12, 13], Pegasus[14], and Triana [15]. Yu and Buyya[16] present a detailed survey of existing systems and provide a taxonomy which can be used for classifying and characterising workflow systems. We use elements of the taxonomy related to Quality of Service (QoS) and workflow modelling for comparing our approach to other systems. A prominent feature of adding instruments to the Grid is the need for real-time remote control and monitoring of instrumentation. Users and applications will be allowed to make Advance Reservation (AR) of computational resources, storage elements, network bandwidth and instruments. AR guarantees availability of resources and instruments at times specified[17]. In ICENI, the scheduling framework supports AR by using the meta-data of application components together with corresponding historical data stored for the purpose of performance analysis and enhancement. This approach is also used in Pegasus, GridbusWorkflow and Askalon Some systems such as Askalon, DAGMan, ICENI, GridFlow and GridbusWorkflow allow users to define their own QoS parameters and to specify optimisation metrics such as application execution time, desired resources and economical cost. We are using a similar mechanism for performance enhancement and estimation taking into account the real-time QoS constraints.

Worflow languages such as BPEL[4], WSChoreography[5] and YAWL[6] are powerful languages for developing workflows based onWeb Services. However, they do not provide a mechanism for describing QoS requirements. Our approach to overcome this has been to develop a partner language to use with BPEL. Instead of defining a new language for workflows with QoS requirements we use a standard BPEL document along with a second document which points to elements within the BPEL document and annotates this with QoS requirements. This allows us to take advantage of standard BPEL tooling for execution and manipulation as well as provide QoS requirements. As we use XPath[25] notation to reference elements within the BPEL document our approach can be easily adopted to other languages based on XML.

The development and execution of workflows within the Grid is a complex process due to the mechanisms used to describe them and the Web Services they are based upon. The selection of the best resources to use within the Grid is complicated due to its dynamic nature with resources appearing and disappearing without notice and the load on these resources changing dramatically over time. These are issues that the scientist will, in general, not wish to be concerned about. The use of advanced reservations can help to reduce the amount of uncertainty within the Grid. However, in general, the selection of the best service to reserve is a complex process. We adopt the approach of a workflow pipeline presented by McGough et al[22] to decompose this problem into more manageable steps.

Huang et al.[23] have proposed a just-in-time workflow optimisation service in which a workflow calls a proxy service rather than the actual service, which then determines the best resource to use. Our approach improves on this by allowing the end points in a BPEL document to be updated on the fly thus hiding the cost of computing the best service in line with execution. Thus far we have used simple workflow optimization heuristics however we see that our approach can be easily adapted to use more optimal approaches such as stochastic optimisation[24] and overall workload optimisation.

3. Overall Architecture of GridCC

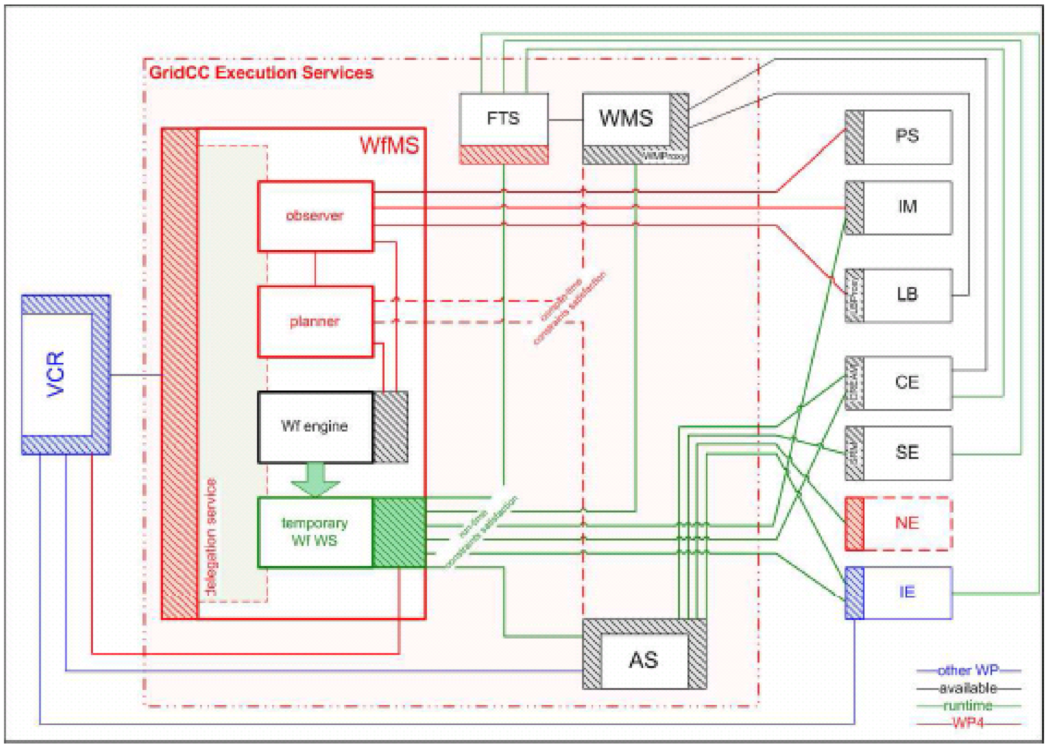

Fig. 1. Overall Architecture of GRIDCC

Figure 1 shows the overall architecture of GRIDCC. In the architecture the Virtual Control Room (VCR) is the user interface to the Grid. The Workflow Editor is inter grated within the VCR. The Workflow Management Service contains the Workflow Planner, which is the main component that performs the QoS functions, and Observer along with the Workflow Engine, in the case of GRIDCC this is the ActiveBPEL [26] engine. The Workflow Engine may communicate with various Grid services such as Compute Elements (CE), Storage Elements (SE) and, as defined within the GRIDCC project, the Instrument Elements (IE) C a Web Service abstraction of an instrument along with the Agreement Service (AS) for brokering reservations with other services. The Performance Repository (PR) contains information about previous executions of tasks on the Grid.

On receiving a workflow and QoS document the WfMS must decide, based on information from the PR along with an understanding of the workflow, if the submitted request can be achieved within the provided QoS constraints. In order to achieve this the WfMS may choose to manipulate the workflow. This may include making reservations for tasks in the workflow and/or changing the structure of the workflow. The Workflow Engine is invoked and is responsible for the ordered execution of the workflow tasks, with the Observer monitoring progress.

4. Workflow with Quality of Service Requirements on Grid

Quality of Service (QoS) is a broad term that is used in this context to denote the level of performance and service that a given client will experience at a specific point in time, when invoking a given specific operation on a Web service instance. QoS support refers to the possibility that a service instance is capable of offering a performance level which satisfy the requirements of a client. QoS is particularly relevant to many of the Grid applications, which are based on collaborative tools and on the real-time interaction with Instrument Elements.

QoS support is paramount given the inherent shared nature of Grid services, and the limited capabilities (in terms of hardware and software resources) that are typically available to satisfy a client's request. Especially for those more desirable services. In this environment, an aggressive best-effort usage of services can consequently cause denial of service problems.

We believe that the possibility of supporting client requirements according to two complementary approaches:

- **loose (soft) guarantees:** The provisioning of loose guarantees consists of the capability of clients, to select service instances that can potentially provide a best-effort QoS profile meeting the client's requirements. This is based on previous measurements of that service on similar machines. From these previous measurements it may be possible to estimate performance values for the service. Loose guarantees are delivered on a best-effort basis, and for this reason, they generally do not require the prior establishment of a Service Level Agreement (SLA) and the consequent negotiation of a contract. Loose guarantees are suitable to those application that can perform adaptation and self-healing operations to cope with requirements that are not met in practice e.g. When a client can simply adapt by switching to an alternative service, or for situations where a statistical approach to QoS is sufficient e.g. 60% of the sensors must switch on with 20s.
- **strict (hard) guarantees:** Strict QoS requires the certainty of the delivery of the prescribed level of service, which needs to be agreed upon through signaling and negotiation processes involving both the client and the service provider. Strict guarantees require an enforcement through fabric-layer mechanisms that are servicespecific. Reservation of a given amount of resources (for example, RAM memory, disk space, network bandwidth, etc.) is one of the most popular mechanisms for the support of strict guarantees. The reservation service provider is responsible of keeping information about resource availability over time, of ensuring that the total amount of resources allocated does not exceed the maximum amount of resources that can actually be locked, and of supporting resource-specific locking mechanisms to guarantee exclusive access to reservation users.

QoS provisioning of our work relies on both strict and loose QoS guarantees. While hard QoS requires the making of reservations on the resources to be used, soft QoS requires the user (or planner acting on the users behalf) to model the execution of the services they require. These models may vary from from the simple, when only a single service is required to the extremely complex when several services are required as part of a workflow. In such cases it may be required that a reservation is required even to satisfy loose QoS constraints.

5. A QoS Aware WfMS

Figure 2 illustrates in detail the internal structure of the WfMS. After users' producing a BPEL4WS document and a QoS document using workflow editor. Due to the limited paper space, we only try to show how QoS functions are carried by our WfMS, we will only explain the WfMS components (workflow editor, workflow planner) that are directly related to QoS.

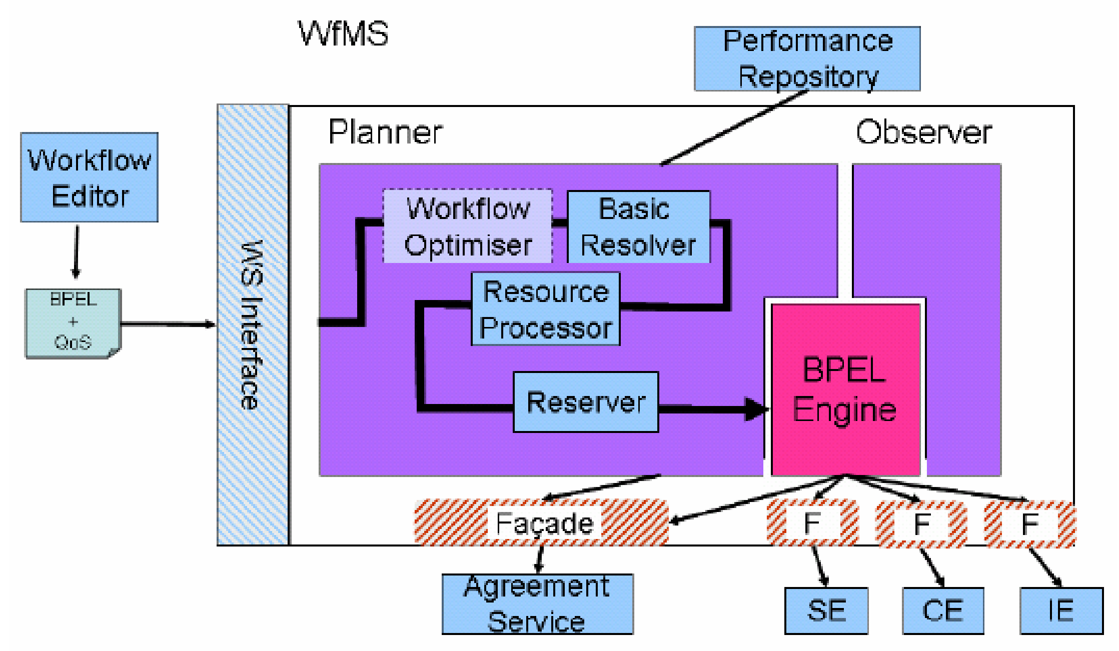

Fig. 2. Components of QoS Aware Workflow Management System

5.1 Connecting BPEL4WS Documents with QoS Requirements

As all the components are chained by BPEL4WS and QoS documents that are passed through, we will explain how this two documents are connected together before we get into the details of individual QoS component. An user submits a BPEL document, which is likely to contain a number of service invocations-referred to as a task.Without loss of generality this document may contain one or more tasks. A separate document is used to describe the QoS requirements placed onto this BPEL document. A QoS requirement document is composed of sets of QoS constraints, which are linked to separate BPEL activities (both basic activities and structure activities) as shown in Figure 3

QoS requirements can further be classified into four categories according to the range **Fig. 3.** Connecting QoS Document and BPEL Model of activities in BPEL models that it specifies:

– **Global requirement** is a QoS document that specifies single global QoS requirements. Thus everything within a BPEL4Ws documents must match these QoS requirements.

– **Single invoke activity requirement** is a QoS document that specifies requirement on a particular BPEL invoke activity. Only that invoke activity needs to satisfy the QoS requirement specified.

– **Multiple invoke activities requirement** is a QoS document that specifies requirement

on a set of invoke activities. In a single QoS constraint, several *XpathReferences* pointing to different invoke activities in a BPEL model are defined.

– **Separate QoS requirements** specifies several QoS constraints elements which might be independent on each other in a QoS document.

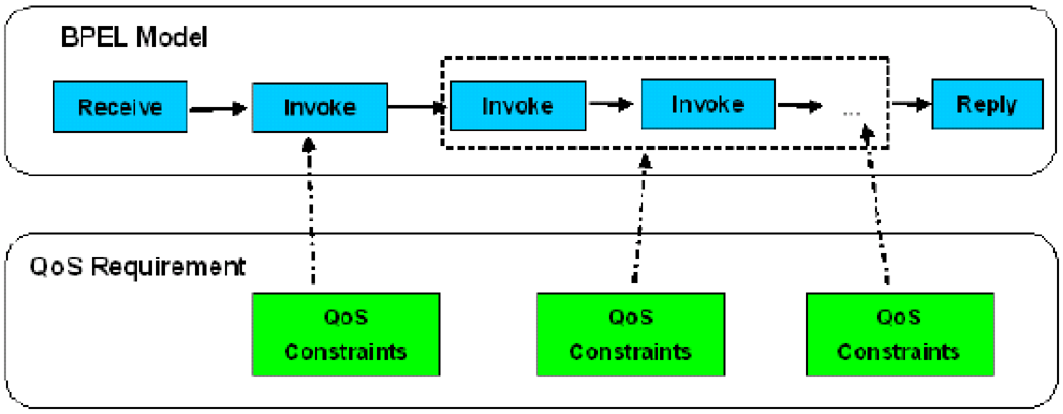

Fig. 3. Connecting QoS Document and BPEL Model

5.2 Workflow Editor

The flexibility and richness of features in the BPEL4WS specification, brings unavoidable complexities. These complexities hinder the use of standard workflow by scientist and researchers in academic domain. Also, as what can be seen from the previous section, writing Qos requirements and attaching them with corresponding BPEL documents manually are tedious and error prone. In order to facilitate the production of the input documents (BPEL and QoS) of our QoS aware WfMS, we provide user a workflow editor. Although, a number of BPEL4WS editors exist, such as Active BPEL Editor and Oracle BPEL4WS Process Manager, these are developed for computer scientists who wish to develop workflows and have an extensive knowledge of BPEL, and WSDL etc.. Our workflow editor differs from existing editors in the following aspects:

– Portal Based Editor: Our JSR 168[19] compliant workflow editor will provide the editing tool on demand. Users can edit and save the workflow on the server and can access them from anywhere and whenever required. Use of JSR 168 complaint portal and portlet allows mixing the presentation layer of the editor with the back end business logic implemented in Java. Browser based clients have the inherent advantage as they do not need to be upgraded on the client side and provide a pervasive access mechanism.

– Drag and Drop: Our workflow editor provides a drag and drop facility to drag various BPEL4WS activities, Web services or operations from Web service registry, Quality of Service (QoS) constraints from QoS panel and variables from XML Schema registry into workflow designer pane.

– Hiding Complexities: A generic BPEL4WS workflow designer demands advanced skill from workflow designers; i.e. thorough understating of Web Services architecture and different types and styles ofWeb Services, expertise in managing XML files from various namespaces, experience of any XML query specification such as XPath or XQuery; and familiarity with the BPEL4WS specification. Web Services and BPEL4WS specification have dependencies on other related specification e.g. WS-Addressing; which further complicates the designing process. The GRIDCC editor hides these complexities from the scientist and researchers by automating different tasks.

The snapshot of our QoS supported workflow editor is shown in Figure 4.

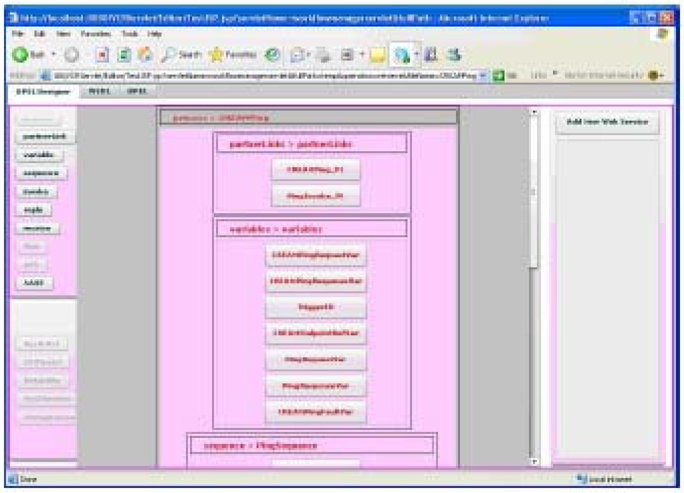

Fig. 4. QoS Supported Workflow Editor

5.3 Performance Repository(PR)

Central to being able to perform any quality of service decisions, whether they be for hard or soft QoS, is information about the resources available. This information needs to be available. The place where this information is stored called the Performance Repository(PR). The PR will serve data of three generic types:

- ♦ Static information about how a particular service scales according to factors such as input, expected output and the task that it is running. This scaling information will be also be as function of the esource on which the service is running. For a web service the information may be broken down into different parts of the process of running the service (de-serialisation of the input message, time for the service to run, serialisation of the output message etc) and then stored in this form.
- ♦ Dynamic information about the current state of the resources on the grid. For conventional Grids (such as the EGEE Grid) this will simply be a redirection from the existing IS however for IEs it may involve having listeners subscribed to the IMS of the IE. An important set of information will be on the current networking status between two sites.
- ♦ Eventually, the PR will also be able make predictions about future usage. Initially this will be through using knowledge of advanced reservations and ultimately using tools such as Network Weather Service.

There are two customers for the PR's information, the individual user and the WfMS planner, and both use it in an essentially the same ways. When making a reservation the PR is interrogated in order to acquire a list of the resources available that can satisfy the QoS demands. For soft QoS constraints the PR provides the input data to the models used to determine whether or not it is possible to manipulate the workflows in such a way as to realise the QoS constraints.

5.4 Workflow Planner

The workflow planner is responsible for ensuring QoS adherence. This is achieved through the use of a number of stagesCsee Figure 2 namely *Workflow optimiser Basic Resolver, Resource Processor and Reserver*. These components are also chained by the QoS and BPEL4WS documents that are passed through. We explain each of them in detail in the following sub-sections.

Workflow Optimiser This is an optional component that provides optimising functionality into the planner. No performance or resource information is used by this component. What it does is to optimise the original input workflow through various of approaches:

- Re-ordering of components: Since workflows (often composed from composite workflows) may contain non-optimal ordering of components, re-ordering of components might be necessary using some of domain knowledge in order to improve the workflow performance.
- Remove redundant components: Workflow may contain unused components, especially when it is composed from other sub-workflows. Thus, redundant components have to be removed.
- Substitution of components: Certain components of defined workflow might be unavailable and have to be replaced by those available or those with better performance.

Various of techniques can be adopted for workflow optimiser and currently, this component is completely optional as addressed earlier. But we believe it can be seamlessly adopted in our WfMS while it’s necessary.

Basic Resolver: The first component that processes QoS requirements of the workflow planner is basic resolver. Given a QoS document which states only descriptions of the unnamed resources needed or states that a particular named resource has been reserved in advance, basic resolver will query the performance repository either to select lists of resources endpoints that matches users’ QoS resource description or to retrieve the information of the named resource defined. If the information retrieved from PR doesn’t match the description described by users for the named resource, the whole workflow will be rejected to user, otherwise the the submitted workflow and a list of resources’ endpoints fetched from PR(if there is any) will be be passed to the next component(*Resource Processor*) for further processing. The QoS element requesting a reservation without a named resource is changed into an element requesting a reservation with a named resource. This can then be passed onto the QoS Reserver for making the actual reservations. No inspection of the BPEL document is done at this stage.

Resource Processor Resource processor plays with the received resources information from basic resolver. It reduces the number of resources that are considered for use. Its implementation is based on a constraints equation method. The workflow along with the QoS requirements is converted into a set of constraint equations. Information from the Performance Repository for the resources is used to solve these constraint equations. Different utility functions can be adopted according to different users’ requirements for selecting the best resources to solve the constraint equations.

Reserver The QoS reserver inspects incoming QoS documents looking for requests for making reservations with known resources. The Agreement Service is contacted in order to make these reservations. The QoS document is then updated to indicate that the reservation has been made and records the unique token used to access the reservation. All requests for reservations are processed here. In the current implementation if reservations cant be satisfied then the whole document will be thrown back to the user to select new reservation times. Once we have components capable of selecting timings for reservations internally the workflow and QoS will be returned to this component.

6. Conclusion and Future Work

In this paper we have presented how the function of quality of services are used within the GRIDCC project to allow for integration of existing Grid technology with that of instruments. Workflows are defined through an editor which allows the augmentation of QoS requirements, defining the users expectations for the execution. The WfMS provides a mechanism for building QoS on top of an existing commodity based BPEL4WS engine. Thus allowing us to provide a level of QoS through resource selection from apriori information along with the use of advanced reservation. The workflow editor and Observer are areas of current development within the project. We are working with application scientists from the GRIDCC project to abstract the editor from the BPEL/QoS languages and make them more accessible to the scientists. We are investigating other techniques which will allow us to dynamically change the execution of the BPEL workflow once deployed to the engine.

References

[1]. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure.*Morgan Kaufmann, July 1998.

[2]. D. Nickull and F. McCabe. Soa reference model. http://www.oasis-open.org/committees/tchome.php?wgsoa-rm.

[3]. W3C. Web Service. http://www.w3.org/TR/ws-arch/.

[4]. T Andrews and F Curbera and H Dholakia and Y Goland and J Klein and FLeymann and K Liu and D Roller and D Smith and S Thatte and ITrickovic and SWeerawarana. Business Process Execution Language forWeb services version 1.1, (BPEL4WS). http://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, May 2003.

[5]. David Burdett and Nickolas Kavantzas. http://www.w3.org/TR/ws-chor-model/.

[6]. W.M.P. van der Aalst and L. Aldred and M. Dumas and A.H.M. ter Hofstede. *Design and Implementation of the YAWL system.* In Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia, june 2004. Springer Verlag.

[7]. D.J. Colling, L.W. Dickens, T. Ferrari, Y. Hassoun, C.A. Kotsokalis, M. Krznaric, J. Martyniak, A.S. McGough, and E. Ronchieri. *Adding Instruments and Workflow Support to Existing Grid Architectures.* volume 3993 of Lecture Notes in Computer Science, pages 956C963, Reading, UK, April 2006.

[8]. T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. S. Jr, and H. L. Truong. *ASKALON: a tool set for cluster and Grid computing.* Concurrency and Computation: Practice and Experience, 17(2-4):143C169, 2005.

[9]. 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.

[10]. J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. *GridFlow: Workflow Management for Grid Computing.* In Proceedings of 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan. IEEE CS Press, Los Alamitos, 12C15 May 2003.

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

[12]. A. Mayer, S. McGough, N. Furmento,W. Lee, S. Newhouse, and J. Darlington. *ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time.* In UK e-Science All Hands Meeting, Nottingham, UK, pages 894C900. IOP Publishing Ltd, Bristol, UK, Sep. 2003.

[13]. S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. *Workflow Enactment in ICENI.* In UK e-Science All Hands Meeting, Nottingham, UK, pages 894C900. IOP Publishing Ltd, Bristol, UK, Sep. 2004.

[14]. E. Deelman, J. Blythe, Y. G. C. Kesselman, G. M. andK. Vahi, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. *Mapping Abstract Complex Workflows onto Grid Environments.*Journal of Grid Computing, 1(1):9C23, 2003.

[15]. I. Taylor, M. Shields, and I. Wang. *Resource Management for the Triana Peer-to-Peer Services.* In J. Nabrzyski, J. M. Schopf, and J. Weglarz, editors, Grid Resource Management-State of the Art and Future Trends, pages 451462. Kluwer Academic Publishers, 2004.

[16]. J. Yu and R. Buyya. *A taxonomy of workflow management systems for grid computing.* GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005.

[17]. S. Andreozzi and T. Ferrari and S. Monforte and E. Ronchieri. *Agreement-based Workload and Resource Management.* In Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, Dec. 2005. IEEE Computer Society.

[18]. Xml path language (xpath) version 1.0. Technical report, 1999. 19. Alejandro Abdelnur and Stefan Hepper. Porlet specification (jsr-168). http://jcp.org/aboutJava/ communityprocess/review/jsr168/.

[20]. Gary Grossman and Emmy Huang. ActionScript 3.0 overview. http://www.adobe.com/devnet/ actionscript/articles/actionscript3 overview.html, June 2006.

[21]. Christophe Coenraets. An overview of MXML: The Flex markup language. http://www.adobe.com/ devnet/flex/articles/paradigm.html, Mar. 2004.

[22]. A. McGough, J. Cohen, J. Darlington, E. Katsiri, W. Lee, S. Panagiotidi, and Y. Patel. *An End-to-end Workflow Pipeline for Large-scale Grid Computing.* Journal of Grid Computing, pages 1C23, Feb. 2006.

[23]. L Huang and DW Walker and Y Huang and and OF Rana. *Dynamic Web Service Selection for Workflow Optimisation.* Proceedings of the UK e-Science All Hands Meeting, Sept. 2005.

[24]. Y Patel and AS McGough and J Darlington. *QoS Support for Workflows in a Volatile Grid.* In Procedings of the 7th IEEE/ACM International Conference on Grid Computing, Barcelona, Spain, Sept. 2006.

[25]. Xml path language (xpath) version 1.0. Technical report, 1999.

[26]. A. endpoints. ActiveBPEL Engine (2.0). http://www. activebpel.org.