Achieving Real-Time in Distributed Computing:

From Grids to Clouds

Dimosthenis P. Kyriazis
National Technical University of Athens, Greece

Theodora A. Varvarigou
National Technical University of Athens, Greece

Kleopatra G. Konstanteli
National Technical University of Athens, Greece

Information Science
REFERENCE

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Achieving real-time in distributed computing: from grids to clouds / Dimosthenis P. Kyriazis, Theordora A. Varvarigou, and Kleopatra Konstanteli, editors.
p. cm.
Includes bibliographical references and index.
Summary: "This book offers over 400 accounts from a wide range of specific research efforts on methodologies, tools, and architectures for complex distributed systems that address the practical issues of performance guarantees, timed execution, real-time management of resources, synchronized communication under various load conditions, satisfaction of QoS constraints, as well as dealing with the trade-offs between these aspects"-- Provided by publisher.
ISBN 978-1-60960-827-9 (hardcover) -- ISBN 978-1-60960-828-6 (ebook) -- ISBN 978-1-60960-829-3 (print & perpetual access) 1. Real-time data processing. 2. Electronic data processing--Distributed computing. I. Kyriazis, Dimosthenis P., 1973- II. Varvarigou, Theordora A., 1966- III. Konstanteli, Kleopatra, 1981-
QA76.54.A328 2011
004'.33--dc23
2011016061

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 7
Workflow Management Systems in Distributed Environments

Spyridon V. Gogouvitis
National Technical University of Athens, Greece

Kleopatra G. Konstanteli
National Technical University of Athens, Greece

Dimosthenis P. Kyriazis
National Technical University of Athens, Greece

Gregory Katsaros
National Technical University of Athens, Greece

Tommaso Cucinotta
Scuola Superiore Sant'Anna, Italy

Michael Boniface
University of Southampton IT Innovation Centre, UK

ABSTRACT

With the advent of Service Oriented Architectures, more applications are built in a distributed manner based on loose coupled services. In this context, Workflow Management Systems play an important role as they are the means to both define the processes that realize the application goals as well as implement the orchestration of the different services. The purpose of the chapter is to give an overview of various solutions regarding workflow semantics and languages, as well as their enactment within the scope of distributed systems. To this end, major focus is given to solutions that are aimed at Grid environments. Scheduling algorithms and advance reservation techniques are also discussed as these are among the hottest research topics in Workflow Management Systems.

INTRODUCTION

The recent technological advances in the field of computing and networking have led to the rapid growth of the internet. People around the world started using it as a tool for not only information exchange, but also as a way to perform various tasks, from online shopping to working remotely. Enterprises were forced to keep up with the internet boom in order to increase profits both by cutting costs and fulfilling the drastically increasing demands of the customers. Organizational efficiency and responsiveness are critical factors in determining the profitability of an organization, especially in the today's global business environment, where resources and people tend to be geographically

DOI: 10.4018/978-1-60960-827-9.ch007

dispersed. The same can be said for the eScience community, where profit takes on a different form. For these reasons a process oriented view is used to formalize the business processes that take place within an organization, be it in the business or the science domain. A Business process is defined as (Coalition, Terminology, & Glossary, 1999) "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer". In essence a business process or workflow defines who needs to do what and in what order in order to achieve a specific goal. With the emergence of distributed environments such as grid and cloud infrastructures, which provide many benefits such as reliability, cost-effectiveness and scalability, workflow management systems have received great attention. Workflow management constitutes to the formalization of a business process and its automatic enactment. This provides many advantages since components that are responsible for executing a specific task can be autonomously developed without the need to incorporate any business logic within them. This task is left to the workflow management system. This also means that components can be reused in different workflows, while workflow can be easily changed in order to provide new functionalities. In this chapter we aim to define the core characteristics of workflow management systems and review some of the solutions proposed within the context of distributed environments.

BACKGROUND

The term 'workflow' is used with various meanings depending on the domain in which the term is used. The Workflow Management Coalition's definition is based on document oriented business processes: "*Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*" (Coalition, et al., 1999).

In the Grid community, the term workflow is typically used in the context of electronic services that may, or may not, be distributed. For example "*Workflow is a pattern of business process interaction, not necessarily corresponding to a fixed set of business processes. All such interactions may be between services residing within a single data centre or across a range of different platforms and implementations anywhere*" (Treadwell, 2005).

In the eScience community, workflow is often used to refer to the use of workflow techniques to support the scientific process, i.e. for performing the activities that take place as part of scientific endeavour in a structured, repeatable and verifiable way. For example, in bioinformatics the scientific process can involve the use of 'in silico experiments', where local and remote resources to test a hypothesis, derive a summary or search for patterns (R. Stevens et al., 2003).

In the Business Process Management and Web Services domain, the term 'workflow' tends to mean programming and automation of processes that involve software exposed as services. This is applied in a variety of areas, e.g. enterprise application integration, supply chains, and business process automation.

WORKFLOW MANAGEMENT SYSTEMS

Overview

Traditionally, information systems have been implemented without the explicit consideration of a business process. Discreet components within a system need to be operated manually and their results propagated to the next component within the process chain. This has also led to the incorporation of process logic within application programs. While this solution can make a system work, it has many drawbacks. Management of the system becomes difficult and costly, as internal changes to any component may have an effect on others. It also lowers interoperability with other systems, thus hindering integration of new components or processes.

Service Oriented Architecture (SOA) (Erl, 2005) is an architectural style that emphasizes implementation of components as modular services that can be discovered and used by clients. Infrastructures based on the SOA paradigm are called Service Oriented Infrastructures (SOIs). Managing the application workflow operations within a SOI requires the orchestration of the distributed resources (Mayer et al., 2004). In that frame, workflow is an important factor for application composition promoting inter-organizational collaborations by integrating the teams involved in managing of different parts of a workflow. Besides, literature (Spooner, Cao, Jarvis, He, & Nudd, 2005) describes additional advantages of the workflow management such as the utilization of resources to increase throughput or reduce execution costs and the ability to build dynamic applications which orchestrate these resources. Workflow Management Systems enable the separation of the business process from the application program code. The process is defined in a workflow model both in task and structure level.

A Workflow Management System needs to support the full lifecycle of a workflow which can be divided into two main phases: *Modeling* of the workflow, where it is composed using some language and *enactment*, where the various tasks are executed according to the specification. There exists a multitude of languages for describing workflows, and a corresponding set of workflow engines to enact the defined processes. These range from the high level descriptions more geared toward the human understanding of processes, to the technically oriented approaches focused on the actual automation of the processes.

There are two types of workflow definitions, namely Abstract and Concrete (Deelman, Blythe, Gil, & Kesselman, 2004), (Deelman et al., 2004). In an abstract model, the tasks are described in an abstract form without referring to specific resources for task execution since it provides the ability to the users to define workflows in a flexible way, isolating execution details. In the concrete model, the tasks of the workflow are bound to specific resources and therefore this model provides service semantic and execution information on how the workflow has been composed both for the service instances and for the overall composition (e.g. dataflow bindings, control flow structures). It has to be mentioned that the service instances do not necessarily correspond to resources since within a resource more than one service instances may be available and executable.

A Workflow Management System provides the ability to define, instantiate and enact an application workflow. A Workflow Model / Specification is used to define a workflow, both in task and structure level. An abstract workflow (or workflow template) defines the sets of services to be executed without explicitly mentioning the exact resources needed. In the usually workflow lifecycle, a mapping mechanism is needed which transforms the abstract workflow into an executable one (concrete workflow or workflow instance), that can be used by an enactment engine for the actual execution, as it contains specific resources to be used.

The main aim of a workflow 0management system is to execute a set of tasks according to a predefined order. Apart from this basic requirement there exists a set of requirements that needs to be fulfilled in order to make such a system viable.

Scalability. A WfMS needs to be able to handle multiple concurrent requests for workflow execution without impact on its performance.

Fault handling. Faults are bound to happen both on hardware as well as on software level. The WfMS must firstly be able to acknowledge these and also provide the capability to the application developer to define a set of corrective actions to be taken under certain circumstances.

Declaration of QoS requirements. There are numerous situations where an application user needs to be able to define QoS parameters either for the application workflow as a whole or parts thereof.

Workflow Monitoring. The WfMS must be able to monitor the execution of every running workflow and be able to present the current state to the user.

Security. While security tends to be neglected in such areas as scientific workflows, it is very vital when concerned with business workflows. A WfMS needs to provide the appropriate level of security for all involved parties, having a sound infrastructure for authentication, authorization and secure message exchange between services.

Legacy code support. While most WfMS are targeted towards the SOA paradigm it is also important to be able to execute tasks not developed in a service oriented function. This is feasible by creating service wrappers around legacy code and is important so as to allow for fast integration of legacy application into the SOA universe.

Workflow Semantics and Languages

The most widely used specifications for describing procedural workflows within businesses are XPDL (WfMC, 2005) and ebXML (OASIS, 2006), and the most widely used workflow specification with reference to service oriented architectures is WS-BPEL (OASIS, 2007). Besides the usual orchestration support, WS-BPEL provides mechanisms for specifying business roles and model actual behavior of participants in a business interaction. The W3C is currently working to provide a choreography framework described with WS-CDL (Kavantzas & others, 2005). The focus of BPEL and most business-oriented workflow languages is control flow.

However, extensive research on workflow control patterns has shown that all languages have limitations in terms of what can be easily expressed (Aalst, Hofstede, Kiepuszewski, & Barros, 2003). This insufficient expressivity and lack of rigorous semantics to allow automated checks on correctness and completeness mean that BPEL and similar Web Service workflow standards have limitations when applied to a distributed environment. The work on workflow patterns led Van der Aalst (Van der Aalst) to provide an up-to-date and extensive comparison of workflow languages and implementations (open source and commercial). Van de Aalst has identified specific patterns grouped into control-flow, data, resource and exception handling.

However, it should be noted that workflow standards is a rapidly moving area and the evaluations by the workflow patterns group may, in some cases, lag behind the latest available specifications. For example, WS-BPEL 2.0 was released in summer 2007 and allows processes to be defined along with compensation and exception handlers. Support is already available in Open Source tools, such as the (ActiveBPEL).

Whilst the post-hoc evaluation of existing workflow languages against workflow patterns with well defined semantics allows useful comparison of standards or implementations, it does not address the problem of inherent lack of rigorous machine-interpretable semantics within each workflow language.

The semantic web service community, on the other hand, is producing rigorous models and logics for the semantic description of Web Services from the ground up. Several European projects inc. (SEKT), (DIP), (SUPER) and (ASG) are working together through the European Semantic Systems Initiative (ESSI) and have collaborated to develop the Web Service Modeling Ontology (WSMO) and Web Service Modeling Language (WSML), which has now been submitted for standardization through the W3C. Meanwhile work done by academia and industry through (SWSI) has resulted in the Semantic Web Service Framework (SWSF), which has both a language (SWSL), also submitted to the W3C for standardisation, and an ontology (SWSO) that include a process model. Indeed their complexity and lack of tool support is a barrier to take-up and is a factor that has to be considered when implementing a distributed environment.

At the other end of the spectrum, the practical application workflow technology, in research projects with applications in the engineering and scientific sectors have seen a proliferation of home-grown solutions. These applications are often data and computation intensive and often make use of grid infrastructure and grid workflow technologies. A typical example is the orchestration of compute resources to execute a large scale numerical simulation, e.g. for multi-scale modelling of biological systems (Lloyd et al., 2007). Some solutions are focused on providing support to scientific applications, e.g. Taverna (Oinn et al., 2006), Pegasus (Deelman, Blythe, Gil, Kesselman, et al., 2004) and Kepler (Bowers, Ludascher, Ngu, & Critchlow, 2006), while others are addressing more architectural issues like KWF-Grid (Neubauer, Hoheisel, & Geiler, 2006), Triana (Taylor, Shields, Wang, & Harrison, 2007) and Askalon (Fahringer et al., 2007), or the workflow components in Unicore (Sild, Maran, Romberg, Schuller, & Benfenati, 2005) and (Globus) The approach taken by these projects is often bespoke, including tailoring workflow solutions to specific domains e.g. processing of bioinformatics data, supporting specific requirements not addressed by existing standards e.g. service-to-service transfer of large datasets, or simply to use 'do it yourself' solutions because widely adopted workflow standards tend to lag behind the cutting edge functionality needed in these applications. For example, scientific processes in eScience can often involve user interaction with experiments during the data production, use and archiving lifecycle (Coles et al., 2006).

Whilst many of these projects have developed their own workflow languages and enactment engines rather than adopt existing standards, some have built upon existing formalisms and therefore provide good targets for extension or reuse. For example, Triana is based on Petri-Nets, Akogrimo uses BPEL and the semantic workflow language (OWL-WS) developed in NextGrid is an extension to OWL-S (Beco, Cantalupo, Giammarino, Matskanis, & Surridge, 2005). However, although some use of standards exists, the use of workflow in advanced Grid architectures, e.g. NextGrid and Akogrimo, has yet to see the emergence of a common approach. The consequent waste of effort in reinventing the wheel, the lack of interoperability, and the confusion in the community regarding the best way forward is a recognised problem (Beco, Cantalupo, & Terracina, 2006).

Workflow Scheduling

Workflow scheduling is considered to be a form of global task scheduling as it focuses on deciding where to execute a task and managing the execution of tasks that are inter-depended on shared resources. The resources may be heterogeneous both in terms of local configuration and as well as in terms of policies. The scheduling process should take into account users' QoS constraints in order to satisfy user requirements. Workflow scheduling taxonomy is divided into the following categories:

- *Scheduling architecture*: Workflow scheduling architectures can be divided into three major categories: centralized, hierarchical and decentralized. When a workflow enactment service has a centralized architecture, all scheduling decisions for all tasks in the workflow are taken by one central scheduler. Even though the resulted schedules might be efficient, there are concerns regarding scalability issues. In contrast to centralized scheduling, both hierarchical and decentralized scheduling shares the work among multiple schedulers. In comparison with centralized scheduling, hierarchical and decentralized schedulers are more scalable but this distributed pattern of work may lead to performance degradation for the overall workflow execution.

- *Decision making*: This is the task of mapping workflows onto resources and services. One option is that decision making can be focused on the information of a single task or of a section of the workflow, referred to as local decision making. Alternatively it can take into account information about the entire workflow, referred to as global decision.
- *Planning scheme*: This is the task of translating abstract workflows, i.e. a workflow described in an abstract manner without any reference to specific resources concrete workflows. Workflow planning schemes can be distinguished between static (where the dynamically changing state of the resources is not taken into account) and dynamic (where, in contrast to static, both dynamic and static information about resources is used to make scheduling decisions at run-time).
- *Scheduling strategy*: Workflow scheduling in a distributed system is an NP-complete problem (Fernandez-Baca, 1989). To this end, many heuristics have been developed for near-optimal solutions so as to match users' QoS constraints. Most of the current existing approaches are performance-driven, in the sense that they are focused on minimizing overall execution time. Unlike performance-driven scheduling strategies, market-driven schedulers take into consideration other parameters apart from performance, such as the cost. Trust-driven schedulers base their strategies on levels of trust that depend on security policy, accumulated reputation and attack history etc.
- *Performance estimation*: Performance estimation is one of the most critical tasks associated with workflow scheduling and workflow enactment engines in general. Workflow schedulers use performance estimation techniques to predict the performance of the tasks in a workflow applied on distributed heterogeneous resources, in order to decide on how and where the actual execution will take place. Performance estimation approaches include: simulation, analytical modeling, historical data, online learning, and hybrid.

Fault Tolerance

In a highly dynamic and heterogeneous SOI geographically and organizationally dispersed, heterogeneous resources are incorporated such as computing systems and software, storage systems, instruments, scientific equipment, specialized hardware, communication systems, data sources as well as human collaborators. In such a heterogeneous environment changes are numerous, highly variable and with unpredictable effects. These changes can lead to failure for various reasons: non-availability of required services or software components, overloaded resource conditions, memory shortage, and network fabric failures. For these reasons, workflow management in SOIs should be able to detect and manage failures in order to ensure reliable support of the execution environment.

Workflow failure handling techniques can be divided into two different levels: task-level and workflow-level (Soonwook & Kesselman, 2003). Whereas task-level techniques are concerned with masking the effects the service failures and their impact on the entire workflow, workflow-level techniques are focused on the manipulation of the workflow structure in order to deal with erroneous conditions.

Task-level techniques can be divided into the following categories:

- *Retry*: After failure, the same task is executed again on the same resource.
- *Alternate resource*: After failure, the same task is submitted and executed on a different resource.

- *Checkpoint/Restart*: After failure, task is moved to other resources and maintains its state, so that it can continue its execution from the point of failure.
- *Replication*: The task is executed simultaneously on different resources.

Workflow-level techniques include:

- *Alternate task*: After failure, another implementation of the same task is executed.
- *Redundancy*: Multiple alternative tasks are executed simultaneously on different resources.
- *User-defined exception handling*: Users are responsible for specifying corrective actions for a certain types of failure.
- *Rescue workflow*: This technique ignores failures and continues to execute the rest of the tasks in the workflow, if possible. Afterwards, information that includes statistics about the failures is generated for internal processing by the system and the client.

Advance Reservation

Lack of end-to-end QoS guarantees, especially when working with workflow service-oriented applications, is considered to be one of the most critical impediments to further penetration of SOIs into the industry world since it discourages both Service Providers (SPs) as well as potential clients from making more extended use of the facilities offered by a SOI. Advance reservation of resources is widely considered as a mechanism to address the aforementioned problem and offer to the users the specific QoS guarantees they want (Schwiegelshohn, Yahyapour, & Wieder, 2006). In essence, advance reservation allows the user to request resources from systems for a specific time interval, i.e. start time and end time, and obtain a sufficient number of resources during this time interval to support the execution of the specific job for which the resources were reserved.

Deterministic Advance Reservation

Deterministic advance reservation consists of mathematical methods for reserving resources. It is based on deterministic algorithms for analyzing performance constraints by applying mathematical processes to the various layers and assumptions associated with each candidate node. In general this technique reduces the risk of missing deadlines and failures and increases the overall reliability of the system since they can be run on real-time environments efficiently. However, such methods do have limitations. Existing deterministic scheduling algorithms assume that the parameters for flexibility are static (Farooq, Majumdar, & Parsons, 2005) are mostly focused on minimizing the response time and do not promote resource sharing while in other cases all requests for execution that overlap with previous ones are rejected. Studies on the performance of these mechanisms (Netto, Bubendorfer, & Buyya, 2007) have demonstrated good results in terms of satisfying the given performance constraints. However, it is also shown that they lead in fragmentation of the resources and lower utilization.

Probabilistic Advance Reservation

The basic motivation behind probabilistic advance reservation is the fact that in many cases the input parameters and the run-time conditions are so many, that it is inefficient and sometimes even impossible to analytically examine the entire solution space. Probabilistic methods generally employ the principles behind decision theory and are influenced by prior probabilities that are derived from analytical methods coupled with benchmarking. These prior probabilities are used to determine the posterior probability of an advance reservation decision. These algorithms demonstrate better results in terms of resource utilization and lead to lower costs for the resources, but in general appear to be less reliable than deterministic methods in terms of the resulting performance granted to the running applications.

APPROACHES, IMPLEMENTATIONS AND COMPARISONS

A very interesting and well documented survey is found in (Yu & Buyya, 2005), indicating the characteristics of some of the major players in workflow management systems worldwide. One of the most interesting among them is GrADS (Berman et al., 2001). It is based on the Globus Toolkit, one of the most prevalent Grid middleware, and it aims specifically at applications with large computational and communication load. It supports a Directed Acyclic Graph (DAG) approach for workflows which means that dependencies between tasks are analyzed and their parallelization can be performed for enhanced scheduling purposes. It also supports QoS constraints through estimating the application execution time. This estimation is performed through analytical modelling and historical data. Furthermore it is based on a centralized architecture with both global and local decision making policies, a prediction based planning scheme and a task level rescheduling for fault tolerance purposes.

Another worth to mention implementation is that of Askalon project (Fahringer, et al., 2007), where attention is given to performance oriented applications and is based on the Globus Toolkit too. Performance estimation is also attempted within this project and a positive feature is the decentralized architecture but with a global decision making mechanism.

The GridBus Toolkit (Buyya & Venugopal, 2004) has been developed by the University of Melbourne and is based on market principles. It follows a hierarchical approach and allows the enactment of services that can be found through a grid registry. It supports the definition of QoS constraints, as the user is able to define total cost and task of overall workflow deadline, but not more complex conditions. Gridbus also offers monitoring of resources through the Gridscape plugin, but not for the status of a running workflow. As for fault tolerance the platform provides only the option for alternate resource.

Taverna (Oinn, et al., 2006) is the workflow management system of the myGrid (R. D. Stevens, Robinson, & Goble, 2003) project aimed to supporting data-intensive experiments in molecular biology. The system follows a centralized architecture which poses questions on the scalability of the system. Its scheduling strategy is performance-driven and not combined with estimation of performance. Taverna supports web services as well as programs coded as java classes, but does not provide any QoS guarantees. The system does provide monitoring of running workflows and a friendly environment for the users to manipulate them. As far as fault tolerance is concerned Taverna allows for the definition of either a retry operation or an alternate location of the same service.

In ICENI (S. McGough, Young, Afzal, Newhouse, & Darlington, 2004), another GT oriented implementation with a computationally intensive scope, the estimation is less efficient than the previous ones but it has a more market driven approach, with a global decision making policy, centralized architectural scheme and a varying scheduling strategy. In all of the above cases a very positive feature is that the workflow composition system can be extended by the user.

Pegasus (Deelman, Blythe, Gil, Kesselman, et al., 2004) is the workflow manager of GriPhyN (Deelman, Blythe, Gil, & Kesselman, 2004) and aims to map abstract workflows to grid resources. The abstract workflow is modeled as a Directed Acyclic Graph (DAG) containing logical files and application components and before being mapped to the actual resources an optimization step takes place. Pegasus is based on a centralized architecture and does not support QoS specifications or monitoring during the execution of a workflow. For fault tolerance Pegasus provides a remapping of an entire subworkflow to new resources.

UNICORE (Forum, 2003) is an approach which is becoming quite popular lately, mainly due to its extremely robust nature. It is mostly based on internal components rather than already circulating solutions for Grid integration and it comes with a number of disadvantages such as concrete workflow models. In a concrete model the user specifies which part of the workflow must be executed in a specific resource. This feature not only limits scheduling performance, as in the case when many users want to access a single resource while others are idle, but also is quite risky in dynamic environments where resources come and go unexpectedly. Nevertheless, one popular characteristic of UNICORE is that many of its mechanisms such as decision making, planning policy and strategy are designed in order to be extensible by the user. The workflow is modeled as a directed acyclic graph with temporal dependencies, even thought advanced flow control constructs, such as *if-then-else* and *repeat until* statements, are supported.

Karajan is part of the Java Commodity Grid (CoG) Kit (Laszewski, Hategan, & Kodeboyina, 2007) which aims to allow Grid users, Grid application developers, and Grid administrators to easily use, program, and administer grids from a high-level framework. The workflow engine, apart from the actual service enacted, interfaces with a visualization component, responsible for providing a representation of the workflow structure as well as monitoring of the execution and a checkpointing subsystem, which provides fault-tolerance to the system. The workflow can be defined in an XML format that provides primitives for generic sequential and parallel execution, sequential and parallel iterations, conditional execution and functional abstraction.

The Kepler workflow system (Ludascher et al., 2006) is an open source application that extends the work of the Ptolemy II (Eker et al., 2003) system to support scientific applications using a dataflow approach. It's main characteristic is that it is based on processing steps, called "actors", which have well defined input and output ports. Users are able to define workflows by selecting appropriate actors and connecting them within a visual user interface. A "director" component holds the overall execution and component interaction semantics of a workflow. The Kepler system provides a various fault-tolerant mechanisms, most important of which is the ability to define actors that are responsible for catching exceptions.

Relevant work in workflow management systems includes (Ming, Xian-He, & Yong, 2006). The authors of the specific paper investigate the effect of resource reservation on external applications as well as on local jobs, and introduce the design of efficient task scheduling algorithms considering the tolerance of local jobs to resource reservation. A new performance metric is proposed, the relative slowdown, to quantify the performance impact of resource reservation. The local job process is modeled with an M/G/1 queuing system and the effect of system parameters on relative slowdown is analyzed. They investigate both first-come-first-serve (FCFS) and round-robin (RR) queuing disciplines. Efficient algorithms are designed and implemented considering local jobs' tolerance to reservation. A user-level soft real-time CPU scheduler, DSRT, is updated to enable resource reservation in a general computing platform. They also define a new metric, the relative slowdown of local jobs on a resource for a given reservation which is the ratio of the average waiting time with reservation and the average waiting time without reservation. DSRT is the implementation of the CPU server of QualMan middleware. A major component of DSRT is the resource scheduler, named the dispatcher. A priority scheduling mechanism is applied to differentiate the processing of real-time (RT) processes and time-sharing (TS) processes. The dispatcher runs at the highest fixed-priority. The approach of the slowdown of running processes as a result of external tasks can be useful in studying the effects of new tasks assigned to resources in comparison with the time constraints.

Paper (Lilan, Tao, Zhanbei, & Minglun, 2004) presents the concept of Manufacturing Grid (MG) as the application of Grid technology. It follows the Open Grid Service Architecture (OGSA) as the system framework, and Globus Toolkit 3.0 (GT3) as the developing toolkit. A QoS-based Global Process Planning (GPP) and scheduling schema (Manufacturing Grid Resource Scheduling, MGRS) is presented and the corresponding module is implemented with the functions of GPP analyzing, resource discovery, AHP (Analytic Hierarchy Process)-based resource selection and fault-tolerant handling (re-scheduling). The goal of the schema is to be the supplement of the application Grid in manufacturing with better QoS performance, such as higher user satisfaction, product quality and service, as well as the lower failure rate, time-to-market, cost, etc. Its main features are:

- *GPP Analyzing*: Refers to analyzing the submitted task, and decomposing the complicated target task into a few serial or parallel simple, basic manufacturing subtasks, according to its QoS properties
- *Resource Selection*: Once the list of possible resources is known, MGRS will select a resource that is expected to meet the requirements mostly. In order to fulfil the restrictions, it has to gather dynamic information about resource accessibility, machining precision, machining capability, and resource status, etc
- *Fault-tolerant Handling*: This is also called re-scheduling. Manufacturing Grid is inherently a dynamic system where environmental conditions are subjected to unpredictable changes as: resource or network failures, system performance degradation, variations in the cost of resources, etc. Fault-tolerant handling is the efficient way to guarantee that the submitted tasks are completed and that the user's requirements are met.

AHP is particularly useful for evaluating complex multi-attribute alternatives involving subjective criteria. The essential steps in the application of AHP are the following:

- Decomposing in a hierarchical manner, a general decision problem into sub-problems that can be easily comprehended and evaluated,
- Determining the priorities of the elements at each level of the decision hierarchy, and
- Synthesizing the priorities to determine the overall priorities of the decision alternatives.

The most limiting feature is the use of UDDI registry for resource discovery. As already mentioned previously, UDDI is generally static and lacks monitoring abilities. Moreover GT3 is considered obsolete, as the current working edition is version 4.

The workflow management service within the GRIDCC project (A. McGough et al., 2007) is tasked with optimizing the workflows and ensuring that they meet the pre-defined QoS requirements specified upon them. The project aims at utilizing instruments through Grid infrastructures, which means that there are serious real-time issues to be met. It also focuses on Web Services and SOA and implements a partner language to use with BPEL4WS. Instead of defining a new language for workflows with QoS requirements, or embedding QoS requirements within a language such as BPEL4WS, GRIDCC uses a standard BPEL4WS document along with a second document which points to elements within the BPEL4WS document and annotates this with QoS requirements. This allows to take advantage of standard BPEL4WS tooling for execution and manipulation as well as to provide QoS requirements. XPath notation to reference elements is used within the BPEL4WS document; therefore, this approach can be easily adapted to other (workflow) languages based on XML. The end-to-end workflow pipeline takes a user's design and implements it within the Grid, through reservation services and performance repository. Workflows are defined through a web based editor which allows the augmentation of QoS requirements by defining the user's expectations for the execution. The WfMS (Workflow Management Service) provides a mechanism for building QoS on top of an existing commodity, based on BPEL4WS engine, thus allowing the provision of a level of QoS through resource selection from a priori information together with the use of advanced reservation.

In the Phosphorus project (Deliverable D.5.2.), a number of scheduling algorithms are proposed in the context of fairness of resource assignment between users and their tasks. An interesting scheme is the fact that Phosphorus takes into consideration the time for completion and the deadline, in order to meet QoS requirements. Phosphorus also deals task workloads, which are either known or not known a priori, and advance reservation. The MetaScheduling Service (MSS) developed by the specific project can tackle complicated workflows allowing the end-user to execute the individual components of his application using the most appropriate resources available. MSS is able to orchestrate resources of different sites belonging to different administrative domains, while it is also responsible for the negotiation of agreements on resource usage with the individual local resource management systems. The main dependency of this implementation is the use of the UNICORE middleware.

In (Brandic, Benkner, Engelbrecht, & Schmidt, 2005) a QoS-aware Grid Workflow Language (QoWL), subset of Business Process Execution Language (BPEL) with QoS extensions and QoS-aware Grid Workflow Engine (QWE) is presented. Users may specify different QoS constraints addressing the overall workflow or individual workflow tasks. QWE comprises of a workflow planning component, which performs QoS negotiation and service selection, and a workflow execution component which executes the workflow by invoking the selected services. Based on the specified QoS constraints, the QWE negotiates with multiple candidate Grid services to select appropriate services which satisfy the specified QoS constraints. Performance prediction for QoS-aware VGE services is based on application-specific performance models which depend on metadata about the input data (e.g. matrix size) supplied by the client during QoS negotiation. VGE services rely on a generic QoS module which usually comprises of an application-specific performance model, a pricing model, a compute resource manager that supports advance reservation and a QoS manager. VGE relies on standard Web Services technologies such as WSDL, SOAP, WS-Security, Tomcat and Axis. VGE services enclose native HPC applications, usually parallel MPI (Message Passing Interface) (MPI) codes running on a cluster, and expose their functionality via a set of common operations for job execution, job monitoring, data staging and error recovery. One limiting factor is the dependency upon application nature.

Workflow has a major role in grid dynamics and is adopted as the core technology for applying grid dynamics in NextGRID. It provides the ability to compose and dynamically adapt grid services available in distributed systems and orchestrate their execution. The workflow components, the services and their environment compose an infrastructure that is described in NextGRID as the concept of Grid Virtual Infrastructure Model or Grid VIM. This infrastructure is designed to allow Grid applications and Grid business models and processes to be combined at run-time. The VIM is designed to provide a run-time adaptable infrastructure, in particular:

- Run-time bindings: workflows need not specify a binding of every task to a specific service, so that the bindings can be chosen at run-time.

- Selective enactment: a single service may provide multiple functions, and it must be possible to choose which is bound to an abstract task, supported by the service.
- Workflow substitution: some abstract tasks may be bound at run-time to more detailed workflows that can be inserted into the enactment at run-time. A common example is substitutions with template business operations such as account and billing workflows.
- Workflow prioritisation. Critical processes, which are either expensive in resources or define the result or the performance of the workflow, must have high priority in the evaluation order.

A key feature of the NextGrid approach is the abstraction of business processes in order for application developers to not have to encode business processes explicitly in their applications. This allows applications to remain functional even if a service provider wants to use a different business model or process (e.g. pay-as-you-go instead of subscription-based access to services). The result is NextGrid's workflow enactment model known as the Grid Virtual Infrastructure Model (Grid VIM) with a corresponding workflow enactment engine. This provides a way to dynamically assemble concrete application workflows using an abstract application workflow specification as a starting point, and introducing business processes at run-time as specified by the service providers and consumers involved in executing the application.

Advance Reservation Implementations

Askalon is a Grid development environment with main objective to simplify the development and optimization of Grid workflow applications. It conducts a performance estimation based both on historical data obtained from a training phase as well as analytical modeling. Furthermore, it uses a variety of predefined as well as user-defined QoS parameters for the discovery process. The approach presented in (Wieczorek, Siddiqui, Villazon, Prodan, & Fahringer, 2006) proposes an extension of Askalon's scheduling service to support advance reservation. The introduced approach performs resource mapping based on a fair-sharing principle that limits the number of resources offered to a single user at a time. Scheduling is based on a list-scheduling heuristic known as the Heterogeneous Earliest-Finish-Time (HEFT) (Topcuoglu, Hariri, & Min-You, 2002) algorithm which is proven to perform well with a low time complexity.

In contrast to most workflow enactment engines, ICENI (S. McGough, et al., 2004) is able to support advanced reservation in some cases. In particular, when it is determined that a resource will be required for a certain interval of time, a reservation is made through a WS-Agreement (*Grid Resource Allocation Agreement Protocol [December 2004]*) procedure. In order to determine the execution start time of each component in the workflow, a repository with historical information about their performance on different resources is used. This information is used during the reservation phase in order to get a prediction of the start time of each component in the workflow. If an agreement to reserve a resource is reached then the components to be deployed to that resource are placed into a queue until the resource reservation time is met. However, this work does not attempt to explore the entire problem space to find the optimal concrete workflow, as this is an NP-hard problem (McMahon & Florian, 1975). Instead, heuristics are used to approximate the optimal solution and a number of existing scheduling algorithms such as random and best-of-n-random have been made workflow aware so that they take the whole workflow into account when scheduling each component. Furthermore, the start time of the workflow executions is strictly defined and the system is responsible for starting the execution of the workflow during the interval of the reservation.

Relevant work includes (Claris, George, & Khaled, 2007) where the authors introduced an implementation of the best-fit scheduling algorithm with deadline constraints for minimizing resource fragmentation so as to employ advance reservation while maximizing utilization. The algorithm reuses concepts of computational geometry to deal effectively with resource fragmentation and deadline requirements, under a generic design that can be adopted to accommodate both network and computing resources. Specifically, it determines whether it is feasible to schedule the job so as to meet its deadline. If so, then it uses a set of criteria to select one of the (possibly multiple) servers who can handle this job, updates its schedule, and returns to the user a reference to this server. Otherwise, the job is dropped. However, the start time of the job is strictly defined and the client can run the job only once per reservation. Furthermore, the scheduling schema presented in this work focuses on trying to minimize the response time instead of maximizing utilization of the resource. Finally, the selection depends on the existence of idle periods between already reserved time periods on each server.

Similarly, the solution offered by the SSS algorithm (Farooq, et al., 2005)consists of periods of continuous utilization of the resource called blocks with idle periods separating the blocks. Scheduled-time of a certain task is the time at which it is scheduled to start its execution under the current schedule of the resource. Given the new request and the current resource schedule, the SSS algorithm accommodates the new request in the resource schedule if it is feasible to do so. The algorithm first identifies all those tasks in the resource schedule that can affect the feasibility of the new schedule with the new request and then tries to work out a feasible schedule for only that subset of tasks. Once this subset of tasks is known, an initial solution can be worked out for that subset and the new request, using any of the well-known strategies such as the earliest-deadline first and the least-laxity-first. However, as in (Claris, et al., 2007) this work does not promote resource sharing and it also considers strict start times of execution.

Netto, Bubendorfer and Buyya in (Netto, et al., 2007) investigate the performance of scheduling with advance reservations. For this purpose they developed a QoS scheduler that uses SLAs to efficiently schedule advance reservations for computation services based on their flexibility. They also introduce the concept of adaptive time QoS parameters, in which the flexibility of these parameters is not static but adaptive according to the user needs and resource provider policies. However, as in (Claris, et al., 2007) and (Farooq, et al., 2005) the scheduling of a job consists on finding a free time-slot that meets the job requirements assuming that the start time of the job is known in advance. Thus, all incoming requests for advance reservation that overlap with any of the previously committed reservations are rejected.

Singh, Kesselman and Deelman in (Singh, Kesselman, & Deelman, 2006) propose a resource provisioning model that describes the resource availability in the form of slots with each slot representing the number of processors available for a certain timeframe at a certain cost. On this basis, they formulate a resource provisioning plan that reserves time slots on resources to optimize a parameterized metric, which combines the economic cost of the resource allocation and the expected application runtime. This resource provisioning planning is combined with two heuristic algorithms, the Min-Min and a genetic algorithm in conjunction with a list-scheduling algorithm. The solution is then tested against workflow-based applications. However, this work deals with optimization as seen from the user's perspective and ignores the utilization of the given resources, which is something that the resource owners are mostly interested in. In addition, this approach does not consider the deadlines that the tasks in the workflow may have.

Zhao and Sakellariou in (Zhao & Sakellariou, 2007) present an advance reservation model for DAG (Directed Acyclic Graph)-based workflows, taking into account the latest possible time that the execution of the whole workflow needs to be completed as specified by the user. Two different approaches are proposed in an attempt to make advance reservations for each task in the DAG by distributing the spare time among the tasks that comprise it. However, this work considers strict start times of execution for the DAGs and ignores resource sharing and maximum utilization.

CONCLUSION

With the advent of service oriented architectures (SOA), applications have started to move away from the monolithic approach towards a paradigm that emphasizes modular design. In such distributed systems workflows play an important role, as there needs to be a high-level component that can orchestrate the different services that make up an application. Most distributed WfMS today are targeted towards a Grid environment and thus have various advantages and disadvantages tied to the Grid ecosystem. In order for the WfMSs to move towards the Cloud paradigm new workflow approaches need to be used that allow dynamic workflow composition and evolution as well as dynamic service discovery and selection during workflow enactment, taking into consideration several QoS parameters. Moreover special attention needs to be paid to the scheduling capabilities a WfMS has to offer. To this end the use of advance reservation mechanisms needs to be researched further. The effectiveness of current advance reservation mechanisms is limited and in many cases not applicable to workflows, mainly due to concerns regarding the dynamic nature of the SOIs and the inability to adjust to the ever-changing resource availability and user demand without obvious consequences on the performance. Furthermore, most of the existing work on advance reservation assumes a precise start time when the workflow starts running, specified either by the user or the advance reservation system itself. Based on this start time, the start times of each task in the workflow are estimated using the end time of the preceding task. The existing studies on relaxed start time advance reservations, namely in cases where the start time of execution is not precise, either assume that the parameters for flexibility are static, or reallocate existing advance reservations at reservation time. At runtime, as the actual deadline approaches, the priority of the job with the flexible start time is increased to ensure that the execution completes prior to its deadline. To overcome these challenges, the mechanisms that perform the task of advance reservation in service-oriented environments should take into account future potential changes in the offered QoS level of a service and provide additional metrics and mechanisms for estimating the QoS level. In addition, such advance reservation mechanisms need to reflect more the real needs of the users by becoming more flexible and relieving the user of strict constrains, such as the start time of the execution and the number of allowed executions per reservation.

REFERENCES

Aalst, W. M. P. V. D., Hofstede, A. H. M. T., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases, 14*(1), 5-51. doi: http://dx.doi.org/10.1023/A:1022883727209

ActiveBPEL. (2010). Retrieved from http://www.activevos.com/ community-open-source.php

ASG. (201). Retrieved from http://asg-platform.org/cgi-bin /twiki/view/Public

Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N., & Surridge, M. (2005). *OWL-WS: A workflow ontology for dynamic Grid service composition* (pp. 148–155). IEEE Computer Society.

Beco, S., Cantalupo, B., & Terracina, A. (2006). *The role of workflow in next generation business oriented Grids: Two different approaches leading to a unified vision*. Paper presented at the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06).

Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., et al. (2001). The GrADS project: Software support for high-level Grid application development. *International Journal of High Performance Computer Applications, 15*(4), 327-344. doi: http://dx.doi.org/10.1177/109434200101500401

Bowers, S., Ludascher, B., Ngu, A. H. H., & Critchlow, T. (2006). *Enabling scientific workflow reuse through structured composition of dataflow and control-flow*. Paper presented at the 22nd International Conference on Data Engineering Workshops.

Brandic, I., Benkner, S., Engelbrecht, G., & Schmidt, R. (2005). *QoS support for time-critical Grid workflow applications*. Paper presented at the First International Conference on e-Science and Grid Computing.

Buyya, R., & Venugopal, S. (2004). *The Gridbus toolkit for service oriented grid and utility computing: An overview and status report.* Paper presented at the 1st IEEE International Workshop on Grid Economics and Business Models, 2004. GECON 2004.

Claris, C., George, N. R., & Khaled, H. (2007). *Efficient implementation of best-fit scheduling for advance reservations and QoS in Grid.* Paper presented at the 1st IEEE/IFIP Intl. Workshop on End-to-end Virtualization and Grid Management (EVGM).

Coles, S. J., Frey, J. G., Hursthouse, M. B., Light, M. E., Milsted, A. J., & Carr, L. A. (2006). An e-science environment for service crystallography - From submission to dissemination. *Journal of Chemical Information and Modeling*, *46*(3), 1006–1016. doi:10.1021/ci050362w

Deelman, E., Blythe, J., Gil, Y., & Kesselman, C. (2004). *Workflow management in GriPhyN- Grid resource management: State of the art and future trends* (pp. 99–116). Kluwer Academic Publishers.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., & Patil, S. (2004). Pegasus: Mapping scientific workflows onto the Grid. In Dikaiakos, M. D. (Ed.), *Grid computing* (*Vol. 3165*, pp. 131–140). Berlin / Heidelberg, Germany: Springer. doi:10.1007/978-3-540-28642-4_2

DIP. (2010). *QoS-aware resource scheduling DIP*. (Deliverable D.5.2., P. P). Retrieved from http://dip.semanticweb.org/

Eker, J., Janneck, J. W., Lee, E. A., Jie, L., Xiaojun, L., & Ludvig, J. (2003). Taming heterogeneity - The Ptolemy approach. *Proceedings of the IEEE*, *91*(1), 127–144. doi:10.1109/JPROC.2002.805829

Erl, T. (2005). *Service-oriented architecture: Concepts, technology, and design*. Upper Saddle River, NJ: Prentice Hall.

ESSI. (2010). Retrieved from http://www.essi-cluster.org

Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., & Nerieri, F. (2007). ASKALON: A development and Grid computing environment for scientific workflows. In Taylor, I. J., Deelman, E., Gannon, D. B., & Shields, M. (Eds.), *Workflows for e-science* (pp. 450–471). London, UK: Springer. doi:10.1007/978-1-84628-757-2_27

Farooq, U., Majumdar, S., & Parsons, E. W. (2005). *Efficiently scheduling advance reservations in Grids*. Ottawa, Canada: Dept. of Systems and Computer Engineering, Carleton University.

Fernandez-Baca, D. (1989). Allocating modules to processors in a distributed system. *IEEE Transactions in Software Engineering, 15*(11), 1427-1436. doi: http://dx.doi.org/10.1109/ 32.41334

Globus. (2010). *Globus Project*. Retrieved from http://www.globus.org

Kavantzas, N., et al. (2005). *Web services choreography description language.* W3C candidate recommendation: 9.

Laszewski, G., Hategan, M., & Kodeboyina, D. (2007). Java CoG kit workflow. In Taylor, I. J., Deelman, E., Gannon, D. B., & Shields, M. (Eds.), *Workflows for e-science* (pp. 340–356). London, UK: Springer. doi:10.1007/978-1-84628-757-2_21

Lilan, L., Tao, Y., Zhanbei, S., & Minglun, F. (2004). *A QoS-based global process planning and scheduling approach for manufacturing Grid*. Paper presented at the In Flexible Automation and Intelligent Manufacturing (FAIM 2004), Toronto, Canada.

Lloyd, S., Gavaghan, D., Simpson, A., Mascord, M., Seneurine, C., Williams, G., et al. (2007). Integrative biology - The challenges of developing a collaborative research environment for heart and cancer modelling. *Future Generations Computer Systems, 23*(3), 457-465. doi: http:// dx.doi.org/10.1016/ j.future.2006.07.002

Ludascher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., et al. (2006). Scientific workflow management and the Kepler system: Research articles. *Concurrency and Computation: Practice and Experience, 18*(10), 1039-1065. doi: http://dx.doi.org/10.1002 /cpe.v18:10

Mayer, A., Mcgough, S., Furmento, N., Lee, W., Gulamali, M., Newhouse, S., et al. (2004). *Workflow expression: Comparison of spatial and temporal approaches*. Paper presented at the Workflow in Grid Systems Workshop, Berlin.

McGough, A., Akram, A., Guo, L., Krznaric, M., Dickens, L., Colling, D., et al. (2007). *GRIDCC: Real-time workflow system*. Paper presented at the 2nd Workshop on Workflows in support of large-scale science, Monterey, California, USA.

McGough, S., Young, L., Afzal, A., Newhouse, S., & Darlington, J. (2004, September). *Workflow enactment in ICENI.* Paper presented at the UK e-Science All Hands Meeting, Nottingham, UK.

McMahon, G., & Florian, M. (1975). On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, *23*(3), 475–482. doi:10.1287/opre.23.3.475

Ming, W., Xian-He, S., & Yong, C. (2006). *QoS oriented resource reservation in shared environments.* Paper presented at the Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006. CCGRID 06.

MPI. (2010). Retrieved from http://www.mpi-forum.org/docs

Netto, M. A., Bubendorfer, K., & Buyya, R. (2007). *SLA-based advance reservations with flexible and adaptive time QoS parameters*. Paper presented at the 5th International Conference on Service-Oriented Computing, Vienna, Austria.

Neubauer, F., Hoheisel, A., & Geiler, J. (2006). Workflow-based Grid applications. *Future Generations Computer Systems, 22*(1-2), 6-15. doi: http://dx.doi.org/10.1016/ j.future.2005.08.002

OASIS. (2006). *OASIS standard v2.0.4- ebXML business process specification schema technical specification* v2.0.4.

OASIS. (2007). *Web services business process execution language*, version 2.0.

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris, J., Glover, K., et al. (2006). Taverna: Lessons in creating a workflow environment for the life sciences: Research articles. *Concurrency and Computing: Practice and Experience, 18*(10), 1067-1100. doi: http://dx.doi.org/10.1002/cpe.v18:10

Schwiegelshohn, U., Yahyapour, R., & Wieder, P. (2006). Resource management for future generation Grids. In Getov, V., Laforenza, D., & Reinefeld, A. (Eds.), *Future generation Grids* (pp. 99–112). Springer, US. doi:10.1007/978-0-387-29445-2_6

SEKT. (2010). Retrieved from http://www.sekt-project.com/

Sild, S., Maran, U., Romberg, M., Schuller, B., & Benfenati, E. (2005). OpenMolGRID: Using automated workflows in GRID computing environment. In Sloot, P. M. A., Hoekstra, A. G., Priol, T., Reinefeld, A., & Bubak, M. (Eds.), *Advances in Grid computing - EGC 2005* (*Vol. 3470*, pp. 464–473). Berlin/Heidelberg, Germany: Springer. doi:10.1007/11508380_48

Singh, G., Kesselman, C., & Deelman, E. (2006). *Application-level resource provisioning on the Grid*. Paper presented at the Second IEEE International Conference on e-Science and Grid Computing.

Soonwook, H., & Kesselman, C. (2003). Grid workflow: A flexible failure handling framework for the grid. *Proceedings of the 12th IEEE International Symposium on the High Performance Distributed Computing*, 2003.

Spooner, D. P., Cao, J., Jarvis, S. A., He, L., & Nudd, G. R. (2005). Performance-aware workflow management for Grid computing. *Comput. J., 48*(3), 347-357. doi: http://dx.doi.org/10.1093/comjnl/bxh090

Stevens, R., Glover, K., Greenhalgh, C., Jennings, C., Pearce, S., Li, P., et al. (2003). Performing in silico experiments on the Grid: A users perspective. *Proceedings of UK e-Science All Hands Meeting,* 2003.

Stevens, R. D., Robinson, A. J., & Goble, C. A. (2003). MyGrid: Personalized bioinformatics on the information Grid. *Bioinformatics (Oxford, England)*, 19.

SUPER. (2010). Retrieved from http://www.ip-super.org/

SWSF. (2010). Retrieved from http://www.w3.org/Submission /SWSF

SWSI. (2010). Retrieved from http://www.swsi.org

SWSL. (2010). Retrieved from http://www.w3.org/Submission /SWSF-SWSL

SWSO. (2010). Retrieved from http://www.daml.org/services/ swsf/1.0/swso

Taylor, I., Shields, M., Wang, I., & Harrison, A. (2007). The Triana workflow environment: Architecture and applications. In Taylor, I. J., Deelman, E., Gannon, D. B., & Shields, M. (Eds.), *Workflows for e-science* (pp. 320–339). London, UK: Springer. doi:10.1007/978-1-84628-757-2_20

The Open Grid Forum. (2004). *Grid resource allocation agreement protocol*.

Topcuoglu, H., Hariri, S., & Min-You, W. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, *13*(3), 260–274. doi:10.1109/71.993206

Treadwell, J. (2005). *Open Grid services architecture glossary of terms*. (Global Grid Forum OGSA-WG. GFD-I, 044).

Unicore Forum. (2003). *Unicore plus final report: Uniform interface to computing resource*. Retrieved December, 2004, from http://www.unicore.eu/ documentation/files/ erwin-2003-UPF.pdf

Van der Aalst, W. (2010). *Workflow patterns*. Retrieved from http://www.workflowpatterns.com

WFMC. (1999). *Terminology and glossary- English*. (Document Number WFMC-TC-1011).

WFMC. (2005). *XML process definition language*. XPDL.

Wieczorek, M., Siddiqui, M., Villazon, A., Prodan, R., & Fahringer, T. (2006). *Applying advance reservation to increase predictability of workflow execution on the Grid*. Paper presented at the Second IEEE International Conference on e-Science and Grid Computing.

WSML. (2010). Retrieved from http://www.wsmo.org/wsml

WSMO. (2010). Retrieved from http://www.wsmo.org

Yu, J., & Buyya, R. (2005). A taxonomy of workflow management systems for Grid computing. *Journal of Grid Computing*, *3*(3-4), 171–200. doi:10.1007/s10723-005-9010-8

Zhao, H., & Sakellariou, R. (2007). *Advance reservation policies for workflows*. Paper presented at the 12th International Conference on Job Scheduling Strategies for Parallel Processing, Saint-Malo, France.