In session "Web Services and Grid Computing," Proceedings of the Conference on Systemics, Cybernetics and Informatics, Orlando, FL, July 2002.

# QoS for Service-oriented Middleware

Amit Sheth, Jorge Cardoso, John Miller and Krys Kochut
Large Scale Distributed Information Systems Lab
Computer Science, University of Georgia
{amit,anton,jam,kochut}@cs.uga.edu, http://lsdis.cs.uga.edu

Myong Kang
Mitretek Systems, Center for Information Systems
myong.kang@mitretek.org

## Abstract

We propose a Service-oriented Middleware (SoM) that provides an upper level of middleware over rapidly emerging Web Services-based middleware to make it easier to develop complex multi-organizational business applications. Our approach primarily builds upon our experience in building distributed workflow management for multi-organizational processes. Effective and efficient Quality of Service (QoS) management is a critical component of SoM allowing it to guarantee the satisfaction and fulfillment of user and application requirements. This paper reviews a QoS model that also supports ability to automatically compute QoS based on QoS specification of component web Services.

**Keywords:** Quality of Service, Web Service QoS, Process QoS, multi-organizational workflows, Service oriented Middleware

## 1 Introduction

Open Grid Service Architecture (OGSA) has presented a vision of an integrated approach to supporting both e-science and e-business (Foster, Kesselman et al. 2002). Related to this is the move towards convergence of formerly distinct communities (Web, Grid and P2P) leading to integrated Internet distributed computing (Hey 2001; Gilmore 2002; Milenkovic 2002). As with most past technological evolution in software architectures, we expect to see a layered approach to achieving the above vision. We propose a Service-oriented Middleware (SoM) that takes a step in this direction by building upon the progress in Grid middleware and rapidly emerging Web Services developed to support business applications. Our objective is to address key outstanding issues such as making it easier to build complex applications defined as processes by composing Web Services, and their management (including orchestration).

Processes based on Web Services are inherently more complicated compared to workflow processes because the scale and heterogeneity is expected to be much greater. Scale will be related to the number of potentially relevant Web service providers that may perform similar jobs and register with multiple registries. Heterogeneity will result from their independent development and modification by providers, resulting in both functional differences such as modeling, and operational differences, even if standards are used.

Through SoM, we aim to provide a higher-level middleware that makes it much easier to build complex applications with the scope of OGSA. The particular classes of middleware services we address as identified in (Chuang, DeFanti et al. 2001) are support for end-to-end QoS objectives, security and robustness. Within the classification of Grid middleware, Grid development environments and tools, and Grid applications and portals (Thomas, Mock et al. 2000), our work primarily belongs to the middle tier.

The services offered by an OGSA needs to include an effective and efficient QoS management, to guarantee the satisfaction and fulfillment of users requirements. For example, when an OGSA is chosen to support e-commerce processes, the architecture must understand the binding agreement or contract between the supplier and customer, specifying QoS items such as products or services to be delivered, deadlines, quality of products, and cost of service. The management of such QoS requirements directly impacts success of organizations participating in e-commerce. Products and services must be available to customers with well-defined specifications to fulfill customer expectation and achieve customer satisfaction.

The OGSA must accept the specification, as well as be able to estimate, monitor, and control the QoS of running applications. QoS can play important role in dynamic scheduling and evolution during enactment (or orchestration) of a process defined as a composition of multiple Web Services. To achieve these objectives the first step is to develop an adequate QoS model for processes. While QoS models have been deployed for various domains, such as networking, multimedia, and middleware, no model has been created for processes. We have investigated adjacent work to decide which dimensions would be relevant to compose a valid and usable QoS model for processes. Based on previous studies and our experience in the workflow domain we have constructed a model composed of the following dimensions: *time*, *cost*, *fidelity*, and *reliability*.

One of the most interesting and important features of the model is that the end-to-end QoS of a composite service or process can be synthesized from the QoS properties of

its components. The QoS properties are a combination of *a priori* estimates from designers as well as estimates computed from prior executions, with the historical data playing a larger role as more data is collected. Synthesizing aggregate estimates requires several problems to be solved, among them the determination of branching probabilities from branching conditions and dealing with correlation between individual services. A QoS model with the properties described here has been prototyped as part of the METEOR workflow management system.

## 2 Background and Relevant Work

First we review recent commercial moves toward service-based architectures and solutions. Then we review our most relevant work in workflow management and semantic interoperability which form the basis of our approach to developing the proposed SoM and its tools. For brevity, we do not review the relevant specifications and standards such as XML, RDF, Web Services (IBM; Graham, Simenov et al. 2002), (including SOAP (SOAP 2002), UDDI (UDDI 2002), WSDL (WSDL 2001) XLANG (Thatte 2001), WSFL (WSFL 2002)), that are the basis of component-based middleware are relevant to OGSA and are also the basis of SoM

**Service-based Infrastructure and Solutions**
The commercial world is moving to service-based infrastructure and solutions. Example of commercial systems and research work based on the service paradigm:

- Propel Platform Development Team (Carey and Team 2001) focuses on a scalable infrastructure for advanced e-services. The Propel team and Anil Nori et al. (Nori, Venketraman et al. 2001) discuss requirements for a comprehensive e-service platform and how the developed system architectures meet them.
- Fabio Casati and Ming-Chien Shan from HP Labs (Casati, Lee et al. 2001)and BizTalk Server 2000 by Bimal Mehta et al. (Metha, Levy et al. 2001) focus on the process-oriented dimension of e-services and discuss how workflow technology contributes to current solutions.
- Vassilis Christophides (Christophides, Hull et al. 2001) from Bell Labs is studying workflow mediation using the XML-based vortex architecture.
- Bell Labs work and the WISE approach by Amaia Lazcano et al. (Lazcano, Schuldt et al. 2001) emphasize process based e-commerce. In their papers, they review ongoing research efforts toward more flexible, interoperable, and highly dependable workflows in an e-service environment.
- The CrossFlow (Grefen, Aberer et al. 2001) is a multi-national research project on cooperation in virtual enterprises. They use a contract mechanism for the service outsourcing and integration.
- Kraiss, et al. (Kraiss, Schoen et al. 2001) discuss the importance of performance guarantees in a banking environment and a mathematical approach for appropriate system configuration.
- Sheth, et al. (Sheth, Aalst et al. 1999) discuss three evolving architectures for multi-organizational processes in the increasingly networked economy.

Additional examples include work by .NET, TIBCO (TIBCO 2002), IBM (IBM), Bowstreet, and many others.

**QoS**
Recent work, much of it funded by DARPA-ITO through the Quorum program, is extending QoS concepts and mechanisms to higher semantic levels to allow the definition, measurement, and control of the quality of service delivered by services and complete applications (Frlund and Koistinen 1998). However, little work has been done in terms of QoS of pluggable components that we propose.

**Workflow Process Management**
Recently, the need for constructing processes across multiple domains using existing applications has received much attention in the context of business-to-business applications (Sheth, Aalst et al. 1999). OGSA has also recognized the need for workflow (Foster, Kesselman et al. 2002). Even though there are many efforts to facilitate such needs, such as enterprise application integration (EAI) and cross-organizational workflow management system (WfMS) (Grefen, Aberer et al. 2001), it is not easy to build complex application, and there is little work that support QoS for end-to-end processes created by composing or coordinating individual applications or Web Services.

The **METEOR** project at the Large Scale Distributed Information systems (LSDIS) Lab in the Computer Science Department of the University of Georgia, has been, perhaps, the largest academic effort (with substantial industrial collaboration). This project has spanned the complete phase of research, prototyping and industrial trials, technology licensing and commercialization of a comprehensive workflow management system. METEOR's architecture includes design tools, monitoring tools, workflow repository, and enactment systems. Due to different needs in organizations we have developed two enactment service: OrbWork (Kochut, Sheth et al. 1999) and WebWork (Miller, Palaniswami et al. 1997; Miller, Cardoso et al. 2002). OrbWork is a CORBA and Java based system oriented to support mission-critical applications requiring high scalability and robustness. It is fully distributed and scalable. Since LSDIS has used Java as the language for its development, the system is portable across platforms. It supports interoperability standards such as jFLOW (JFLOW 1998) and SWAP. Recent enhancements include support for dynamic changes at the instance level (Kochut, Sheth et al. 1999), repository to support process reuse, and an exception handling mechanism (Luo 2000) that is part of the adaptation module. A collaboration with NRL resulted in a extended METEOR system called SALSA to build a *multilevel secure* (*MLS) workflow* management system (Kang, Froscher et al. 1999; Kang, Park et al. 2001).

## 3 Specification of Web Service and Process QoS

Processes are composed of multiple Web services. Selection of services can occur either design time or run time. In either case, matching and ranking functions are used to select one or more suitable services. The service should provide the appropriate functionality as well as meet QoS requirements.

### *3.1 SoM QoS Requirements*

The research and development of mechanisms to specify QoS and allow an effective and efficient QoS management are our main objectives for the SoM architecture. We have identified four important and complementary areas that we are currently investigating: specification, estimation algorithms and methods, monitoring tools, and mechanisms to control the QoS. Only the development of integrated solutions composed of those four modules can result in a sophisticated quality management framework. The objectives and functionalities of each module include the following:

- A QoS model must be developed to allow for the *specification* of workflow QoS metrics. This model allows suppliers to specify the duration, quality, cost, fidelity, *etc.*, of the services and products to be delivered. Specifications can be set at design-time, when designers build workflow applications, or they can be adjusted at run-time.
- Algorithms and methods must be developed to *estimate* the QoS of a workflow both before instances are started and during instance execution. The estimation of QoS before instantiation allows suppliers to ensure that the workflow processes to be executed will indeed exhibit the QoS requested by customers. The analysis of QoS during instance execution allows the SoM infrastructure to constantly compute QoS metrics and register any deviations from the initial requirements.
- Tools must be available to *monitor* the QoS of running applications. Users and managers need to receive information about the QoS status and possible deviations from the desired metrics that might occur. The use of QoS monitoring tools can automatically detect this variation in fidelity and automatically notify interested users.
- Mechanisms must be available which *control* the QoS of applications. Control is necessary when applications do not behave according to initial requirements. Let us consider the following example: workflow instances are running correctly and the QoS specifications are being followed when a task fails. As a consequence, QoS specifications of time are no longer satisfied, and the SoM infrastructure raises a warning, an alert, or an exception. The faulty task needs to be replaced by an equivalent task to restore the soundness of the system. This replacement can be accomplished by applying dynamic changes to the instances, either manually or automatically (Cardoso, Luo *et al.* 2001).

### *3.2 QoS Model*

QoS is typically decomposed into several dimensions. For business processes, (Stalk and Hout 1990) and (Rommel 1995) investigated the features with which successful companies assert themselves in the competitive world markets. Their results indicated that success is related to the capability to compete with other organizations, and it is based upon three essential pillars: *time*, *cost*, and *quality*. These three dimensions have been a major concern for organizations. (Garvin 1988) associates eight dimensions with quality, including performance and reliability. Software systems quality of service has been extensively studied. Major contribution can be found in networking (Georgiadis, Guerin et al. 1996)., eal-time applications (Clark, Shenker et al. 1992) and middleware areas (Hiltunen, Schlichting et al. 2000) (Zinky, Bakken et al. 1997). For middleware systems, (Frlund and Koistinen 1998) present a set of practical dimensions for distributed object systems reliability and performance, which include TTR (time to repair), TTF (time to failure), availability, failure masking, and server failure. For data networks, the QoS generally focuses on domain specific dimensions such as bandwidth, latency, jitter, and loss (Nahrstedt and Smith 1996). Based on previous studies, and our experience in the workflow and process domains we construct a QoS model composed of the following dimensions: *time*, *cost*, *reliability,* and *fidelity*. We will use these four dimensions and develop a QoS framework suitable for Grid/Web services.

In order to be more precise, we provide our definitions of the four dimensions. (1) For a Web service, the *response time* can be defined as the time that elapses between service requests arrival and the completion of that service request. Response time is the sum of waiting time and actual processing time. (2) *Cost* represents the cost associated with the execution of Grid/Web services. It is a fundamental issue for organizations that wish to reduce their expenditures with internal processes and service cost. (3) *Reliability* corresponds to the likelihood that a service will perform for its users when the user demands it and it is a function of failure rate. Reliability is given as the ratio of *successful executions/scheduled executions*. (4) *Fidelity* reflects how well a product is being produced or how well a service is being rendered. Fidelity is treated as a vector composed of fidelity attributes. Each fidelity attribute refers to a property or characteristic of the product being created, transformed, or analyzed. Fidelity attributes are used to determine how well services are meeting user specifications.

### *3.3 Creation of QoS Estimates*

Determining useful estimates for the QoS properties of a Web service can be a challenging task. A combination of *a priori* estimates from designers as well as estimates computed from prior executions will be used, with the historical data playing a larger role as more data is collected. Additional complexities are due to the fact that QoS is parametric. For example, the response time of a service that takes an XML document as input will depend on the size of the document. Estimates for composite Web services can be developed in two ways: (a) estimates

for the entire composite service can be created just like they are for ordinary/atomic services (*i.e.*, *a priori* estimates refined as execution monitoring data is collected), (b) the QoS properties can be synthesized from the QoS properties of the component services making up the composite service. Synthesizing aggregate estimates requires several problems to be solved, among them (1) determination of branching probabilities from branching conditions and (2) dealing with correlation between individual services.

A web service runtime behavior specification is composed of two classes of information: basic and distributional. The basic class associates with each service QoS dimension the minimum value, average value, and maximum value the dimension can take. For example, the cost dimension corresponds to the minimum, average, and maximum cost associated with the execution of a task. The second class, the distributional class, corresponds to the specification of a constant or of a distribution function (such as Exponential, Lognormal, Normal, Rayleigh, Time-Independent, Weibull, and Uniform) which statistically describes task behavior at runtime. The values specified in the basic class are typically employed by mathematical methods in order to compute process QoS metrics, while the distributional class information is used by simulation systems to compute workflow QoS. To devise values for the two classes, appropriate function are applied to derive the QoS metrics for individual services (Cardoso, Miller *et al.* 2002).

### *3.4 Computing Process QoS*

Comprehensive solutions to the difficult problems encountered in synthesizing QoS for composite services are discussed in detail (Cardoso, Luo et al. 2001; Cardoso, Miller et al. 2002). The latter paper presents a mathematical model and a network reduction algorithm for computing aggregate QoS properties step-by-step. At each step, a reduction rule is applied to shrink a process network. This is continued until only one node is left in the network. The set of reduction rules that can be applied to a composite service (network) corresponds to the set of inverse operation that can be used to construct a network of services. To compute QoS metrics, we use a set of six distinct reduction systems: (1) sequential system, (2) parallel system, (3) conditional system, (4) fault-tolerant system, (5) loop system, and (6) network system. As an illustration, we will show how reduction works for a sequence of services.

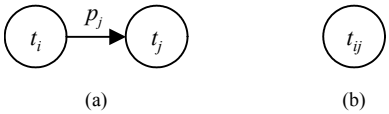


Figure 1 - Sequential system reduction

**Reduction of a Sequential System.** Two sequential service tasks $t_i$ and $t_j$ are reduced to a single task $t_{ij}$. In this reduction the incoming transitions of $t_i$ and outgoing transition of tasks $t_j$ are transferred to task $t_{ij}$.

In a sequential system $p_j = 1$. This reduction can only be applied if the following two rules are satisfied: a) $t_i$ is not a *xor/and* split and b) $t_j$ is not a *xor*/*and* join. These rules prevent this reduction to be applied in a parallel, conditional, and loop systems. To compute the QoS of the reduction the following formulae are applied:

| | |
|---|---|
| Time : | $T(t_{ij}) = T(t_i) + T(t_j)$ |
| Cost: | $C(t_{ij}) = C(t_i) + C(t_j)$ |
| Reliability: | $R(t_{ij}) = R(t_i) * R(t_j)$ |

While mathematical methods can be effectively used, another alternative is to utilize simulation analysis (Miller, Cardoso et al. 2002). Simulation can play an important role in fine-tuning tuning the QoS metrics of workflows, by exploring "what-if" questions. When the need to adapt or to change a workflow is detected, deciding what changes to carry out can be very difficult. Before a change is actually made, its possible effects can be explored with simulation. To facilitate rapid feedback, the workflow system and the simulation system need to interoperate. In particular, workflow specification documents need to be translated into simulation model specification documents so that the new model can be executed/animated on-the-fly. In our project, these capabilities involve a loosely-coupled integration of the METEOR WfMS and the JSIM simulation system (Nair, Miller et al. 1996; Miller, Nair et al. 1997; Miller, Seila et al. 2000). The simulation model is displayed graphically and then executed/animated. Statistical results are collected and displayed, indicating workflows QoS.

### *3.5 System Inplementation*

To enable SoM to support an efficient QoS management several enhancements need to be made to the middleware infrastructure. The enhancements include the development and support of a comprehensive QoS model and the implementation of methodologies (a mathematical model and simulation) to compute and predict QoS. We have developed a stochastic workflow reduction algorithm (SWR) for the step-by-step computation of QoS metrics. Our work has been carried out for the METEOR system to allow the specification, computation, and management of QoS. The support of QoS requires the modification and extension of several workflow system components, and the development of additional modules. While the implementation was made for the METEOR system, and the development was based on a specific conceptual model, the main ideas can be applied to the vast majority of workflow systems available (Aalst, Barros et al. 2002). The support of QoS management requires the modification and extension of most of workflow system components. This includes the enactment system, the workflow builder (or designer), the monitor, the code generator, the repository, the workflow model, and the task model. Additionally, new components need to be implemented, such as a QoS estimator module to create QoS estimates for tasks and probabilities for transitions. The monitor needs an additional interface so that runtime

tasks QoS metrics are propagated and logged into a database for data processing purposes.

## 4 Future Work

**Specification of Process QoS**

Descriptions of QoS for Grid/Web services need to be stored in a fashion suitable for automatic processing. In other words, QoS information should be stored with functional descriptions of services in registries/repositories. As suggested by (Foster, Roy et al. 2001), we are working on defining and using an extended Web Services Description Language (WSDL) to describe services. Similarly, an extended Web Services Flow Language (WSFL) will be used for composite services built out of simpler services.

The extended WSDL should provide additional information prescribed by the OGSA. These include discovery, soft state destruction, explicit destruction, notification source, notification sink, registry, factory, factory primary key, handle mapper, and manageability (Foster, Kesselman et al. 2001). We assume that such descriptions must be provided for a service to be considered a Grid service. We will take a WSDL with extensions for OGSA, and augment it with our additional QoS information with security to form a WDSL+QoS description. These descriptions will be stored in repositories (Arpinar, Miller et al. 2001) that are upward compatible with UDDI registries. Such repositories will allow Grid services be located through advanced query/search mechanisms.

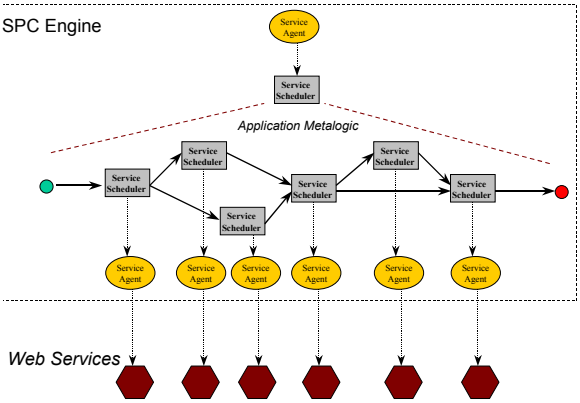


**Figure 2: SPC Engine based on METEOR system**

We also wish to be able to save descriptions of composite Grid services made up of component services. For this will propose to use WSFL+QoS. This specification will indicate the process logic for a composite service. The information is created with our SoM application builder and again stored in the repository. Figure above shows SoM Process Control (SPC) engine, which will be build upon the METEOR architecture. It will be responsible for executing the designed SoM applications.

As we extend WSFL, we will keep track of progress of its competition, namely XLANG (Thatte 2001) from Microsoft and DAML-S from DARPA. In particular, DAML-S (DAML-S 2001) currently includes constructs to specify QoS parameters, such as quality guarantees, quality rating, and degree of quality. While DAML-S has identified specification for Web service and business processes as a key specification component, the QoS model adopted should be significantly improved to supply a more complete solution.

**Maintainability**

The QoS model presented in this paper can be extended in two additional dimensions which are useful for SoM applications with stronger requirements. The first dimension is *maintainability*. Maintainability corresponds to the mean time necessary to repair failures; it is the

average time spent to maintain applications in a condition where they can perform their intended function. Maintenance actions mainly involve the correction of failures during application execution. SoM infrastructures record the period of time necessary for a faulty node to be repaired. The time spent to repair a component depends on the type of error that has occurred. To increase maintainability, advanced mechanisms have been developed to allow workflow infrastructures to automatically recover from errors. Luo *et al.* (2000) describe the architecture and implementation of an exception-handling mechanism. The system detects and propagates exceptions which occur during instances execution to an exception-handling module. The system, based on case-based reasoning theory, derives exception handlers to repair damaged workflows (Luo, Sheth *et al.* 1998). The system has the ability to adapt itself over time. The knowledge acquired in past experiences is used in the resolution of new problems.

**Security**

The second dimension that can be included is the *trust* dimension. The use of workflow systems to coordinate and manage Web-services compels the development of techniques to appraise the global security level of applications specifications and the trust level of the outcome of the overall application. Applications face several security problems, and dedicated mechanisms are needed to increase the level of security. Major problems include the distributed nature of SoM applications, the use of non-secure networks (*i.e.* the Internet), the use of Web servers to access SoM data, and the potential multi-organizational span of SoM. Systems security level is assessed through the existence of security mechanisms (such as authentication, access control, labels, audits, system integrity, security policy, etc.) and through the use of development techniques (such as formal specifications, formal proofs, tests, etc.). The importance of developing secure middleware systems has been recognized, and prototypes combining middleware and security technology have already been developed. We have extended workflow technology with the implementation of two security modules. The first one (Miller, Fan *et al.* 1999) and (Fan 1999) describes a workflow security architecture which targets the five security services (authentication, access control, data confidentiality, data integrity, and non-repudiation) recommended by the International Standards Organization for network-based information systems. The second one (Kang, Froscher *et al.* 1999) describes a multilevel secure (MLS) workflow system to enable distributed users and workflow applications to cooperate across classification levels. MLS workflow systems allow users to program multilevel mission logic, to securely coordinate distributed tasks, and to monitor the progress of the workflow across classification levels.

## 5 References

Aalst, W. M. P. v. d., A. P. Barros, A. H. M. t. Hofstede and B. Kiepuszeski (2002). Workflow patterns homepage, http://tmitwww.tm.tue.nl/research/patterns.

Arpinar, I. B., J. A. Miller and A. P. Sheth (2001). An efficient Data Extraction and Storage Utility for XML Documents. Proc. Of 39th Annual ACM Southeast Conference, Athens, GA.

Cardoso, J., Z. Luo, J. Miller, A. Sheth and K. Kochut (2001). Survivability Architecture for Workflow Management Systems. Proceedings of the 39th Annual ACM Southeast Conference, Athens, GA.

Cardoso, J., J. Miller and A. Sheth (2002). A Quality of Service Model for Workflow Processes. Athens, GA, LSDIS Lab, Department of Computer Science, University of Georgia.

Carey, M. and P. P. Team (2001). "Towards a Scalable Infrastructure for Advanced E-Services." Data Engineering Bulletin 24(1).

Casati, F., S.-M. Lee and Q. Su (2001). "Definition, Execution, Analysis, and Optimization of Composite E-Services." Data Engineering Journal 24(1).

Christophides, V., R. Hull, A. Kumar and J. Simeon (2001). "Workflow Mediation using VorteXML." Data Engineering Bulletin 24(1).

Chuang, J., T. DeFanti, I. Foster, K. Klingenstein, D. Messerschmitt and D. Schmidt (2001). White paper on an NSF ANIR Middleware Initiative.

Clark, D., S. Shenker and L. Zhang (1992). Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. Proceedings of ACM SIGCOMM.

DAML-S (2001). Technical Overview - a white paper describing the key elements of DAML-S.

Fan, M. (1999). Security for the METEOR Workflow Management System. Department of Computer Science. Athens, GA, University of Georgia.

Foster, I., C. Kesselman, J. M. Nick and S. Tuecke (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.

Foster, I., C. Kesselman and S. Tuecke (2001). "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." International J. Supercomputer Applications 15(3).

Foster, I., A. Roy, V. Sander and L. Winkler (2001). End-to-End Quality of Service for High-End Applications.

Frlund, S. and J. Koistinen (1998). "Quality-of-Service Specification in Distributed Object Systems." Distributed Systems Engineering Journal 5(4).

Garvin, D. (1988). Managing Quality: The strategic and Competitive Edge. Free Press, NY.

Georgiadis, L., R. Guerin, V. Peris and K. Sivarajan (1996). Efficient network QoS provisioning based on per node traffic shaping. IEEE ACM Transactions on Networking.

Gilmore, S. (2002). Grid Will Hunting. InfoWorld.

Graham, S., S. Simenov, T. boubez, D. Davis, G. Daniels, Nakamura and R. Neyama (2002). Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, SAMS.

Grefen, P., K. Aberer, H. Ludwig and Y. Hoffner (2001). "CrossFlow: Cross-Organizational Workflow Management for Service Outsourcing in Dynamic Virtual Enterprises." Data Engineering Bulletin Special Issue on Infrastructure for Advanced E-Services 24(1).

Hey, T. (2001). e-Science, e-Business and the Grid. Next Generation HPC Systems and the Grid. Edinburgh, UK.

Hiltunen, M. A., R. D. Schlichting, C. A. Ugarte and G. T. Wong. (2000). Survivability through Customization and Adaptability: The Cactus Approach. DARPA Information Survivability Conference and Exposition (DISCEX 2000).

IBM IBM developerWorks Web Services Zone.

JFLOW (1998). OMG BODTF RFP #2 Submission, Workflow Management Facility, Revised Submission,ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf.

Kang, M. H., J. N. Froscher, A. P. Sheth, K. J. Kochut and J. A. Miller (1999). A Multilevel Secure Workflow Management System. Proc. of the 11th Conference on Advanced Information Systems Engineering, Heidelberg, Germany.

Kang, M. H., J. S. Park and J. N. Froscher (2001). Access Control Mechanisms for Inter-organizational Workflows. Proceedings of 6th ACM Symposium on Access Control Models and Technologies, Chantilly, VA.

Kochut, K., A. Sheth and J. A. Miller (1999). "Optimizing Workflow." Component Strategies **1**(9): 45-57.

Kraiss, A., F. Schoen and e. al. (2001). "Towards Response Time Guarantees for e-Service Middleware." Data Engineering Bulletin 24(1).

Lazcano, A., H. Schuldt, G. Alongo and H. Schek (2001). "Special Issue on Infrastructure for Advanced E-Services." Data Engineering Bulletin 24(1).

Luo, Z. (2000). Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes. Department of Computer Science. Athens, GA, University of Georgia.

Luo, Z., A. P. Sheth, J. A. Miller and K. J. Kochut (1998). Defeasible Workflow, its Computation, and Exception Handling. Proceedings of 1998 Computer-Supported Cooperative Work (CSCW 1998), Towards Adaptive Workflow Systems Workshop, Seattle, WA.

Metha, B., M. Levy, G. Meredith, T. Andrews, B. Beckman, J. Klein and A. Mital (2001). "BizTalk Server 2000 Business Process Orchestration." Data Engineering Bulletin 24(1).

Milenkovic, M. (2002). Peer-to-Peer: Through the Looking Glass, GGF4, February 2002.

Miller, J. A., J. S. Cardoso and G. Silver (2002). Using Simulation to Facilitate Effective Workflow Adaptation. Proceedings of the 35th Annual Simulation Symposium (ANSS'02), San Diego, California.

Miller, J. A., M. Fan, S. Wu, I. B. Arpinar, A. P. Sheth and K. J. Kochut (1999). Security for the METEOR Workflow Management System. Athens, GA, Dept of Computer Science, University of Georgia**:** 33.

Miller, J. A., R. Nair, Z. Zhang and H. Zhao (1997). JSIM: A Java-Based Simulation and Animation Environment. Proceedings of the 30th Annual Simulation Symposium, Atlanta, GA.

Miller, J. A., D. Palaniswami, A. P. Sheth, K. J. Kochut and H. Singh (1997). "WebWork: METEOR's Web-based Workflow Management System." Journal of Intelligence Information Management Systems: 185-215.

Miller, J. A., A. F. Seila and X. Xiang (2000). "The JSIM Web-Based Simulation Environment." Future Generation Computer Systems: Special Issue on Web-Based Modeling and Simulation 17(2): 119-133.

Nahrstedt, K. and J. M. Smith (1996). "Design, Implementation and Experiences of the OMEGA End-point Architecture." IEEE

In session “Web Services and Grid Computing,” Proceedings of the Conference on Systemics, Cybernetics and Informatics, Orlando, FL, July 2002.

JSAC 14(7): 1263-1279.

Nair, R., J. A. Miller and Z. Zhang (1996). A Java-Based Query Driven Simulation Environment. Proceedings of the 1996 Winter Simulation Conference, Colorado, CA.

Nori, A., C. Venketraman and R. Jain (2001). "Defining the Next Generation e-Business Platform: A Discussion of the Asera eBusiness Platform." Data Engineering Bulletin 24(1).

Rommel, G. (1995). Simplicity wins: how Germany's mid-sized industrial companies succeed. Boston, Mass, Harvard Business School Press.

Sheth, A. P., W. v. d. Aalst and I. B. Arpinar (1999). "Processes driving the networked economy." IEEE Concurrency: 18-31.

SOAP (2002). Simple Object Access Protocol.

Stalk, G. and T. M. Hout (1990). Competing against time: how timebased competition is reshaping global markets. New York, Free Press.

Thatte, S. (2001). XLANG: Web Services for Business Process Design, Microsoft, Inc.

Thomas, M., S. Mock and J. Boisseau (2000). Development of Web Toolkits for Computational Science Portals: The NPACI HotPage. Ninth IEEE Intl Symp on High Performance Distributed Computing (HPDC'00), Pittsburgh, PA.

TIBCO, I. (2002). TIBCO and Web Services (Technical White Paper).

UDDI (2002). Universal Description, Discovery, and Integration.

WSDL (2001). W3C Web Services Description Language.

WSFL (2002). WSFL, IBM.

Zinky, J., D. Bakken and R. Schantz (1997). "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems 3(1).