Cardoso, J. and A. Sheth, "Adaptation and Workflow Management Systems", International Conference WWW/Internet 2005, Lisbon, Portugal, 19-22 October 2005. pp.356-364, ISBN: 972-8924-02-X

ADAPTATION AND WORKFLOW MANAGEMENT SYSTEMS

Jorge Cardoso
Departamento de Matemática e Engenharias
Universidade da Madeira
9050-078 Funchal – Portugal
jcardoso@uma.pt

Amit Sheth
LSDIS Lab, Department of Computer Science
University of Georgia
Athens, GA 30602 – USA
amit@cs.uga.edu

ABSTRACT

Today, companies – large and small – can select Workflow Management Systems (WfMSs) to support their business processes. When processes are critical, it is fundamental that WfMS infrastructures continue to provide pre-established service levels to users in the face of disruptions. Adaptation ad-dresses precisely this issue. Current architectures do not incorporate adequate solutions that enhance WfMSs' adaptation. In this paper we present a set of comprehensive techniques to be used in the development of WfMSs to increase their level of adaptation. We discuss how workflow adaptation can be triggered, which adaptation strategies can be applied, and why dynamic changes are in-dispensable to carry out adaptation. We not only target adaptation from a functional perspective, but also from an operational perspective. For the strategies presented, we describe their implementation to the METEOR WfMS.

KEYWORDS

Workflow Management Systems, Adaptation, Quality of Service.

1. INTRODUCTION

Workflow Management Systems (WfMSs), an innovative development of the last decade, represent a fundamental technological infrastructure that efficiently manages business processes. Organizations currently use WfMSs to manage a broad range of distinct applications, such as insurance claims, bank loans, administrative procedures, and healthcare processes. Applications can be more oriented to support or enhance existing processes, to increase competitive advantage, to reduce costs, and also to manage critical infrastructure systems. In many cases, the applications managed by WfMSs are of vital significance to the organizations that govern them. The disruption of services provided by WfMSs could easily incapacitate the completion of the running process. For example, it is not advisable for an insurance company to delay a customer's insurance claims processing due to a WfMS failure. It is also not accept-able to delay a patient's treatment due to a WfMS malfunction.

In reaction to disruptions, adaptive procedures must be undertaken to ensure that the workflow systems continue to meet expected operational requirements. The adaptation research area copes with the disruptions that critical systems face. It has gained the interest of an increasing number of people because our society is becoming more and more dependent on computer systems.

This paper presents a multidimensional solution composed of a set of strategies which, when applied to a WfMS, increases its degree of adaptation. We consider not only functional adaptation, but also operational adaptation. While functional adaptation is triggered when a workflow component ceases its activity, operational adaptation occurs when tasks and workflows do not complete their realization in accordance with

initial quality of service (QoS) requirements. The ideas and strategies presented are applied to workflow systems, but they can be implemented for other distributed systems.

The set of strategies presented in this paper is closely linked to the METEOR WfMS, since this system has been used as a prototype. Thus, section 2 briefly de-scribes the METEOR system and its current implementation status as a prototypical example of today's advanced WfMSs. In section 3 we present the conceptual architecture of METEOR. Our architecture functionally divides the modules that compose a WfMS. This division results in a separation of concepts that allows a clearer vision of both the components that form the WfMS and the areas that need to be considered when implementing adaptation strategies. Section 4 shows adaptation and how adaptive methods can be applied to WfMSs. Related work is discussed in section 5, and section 6 presents our conclusions.

2. THE METEOR WORKFLOW MANAGEMENT SYSTEM

At the LSDIS Lab (Department of Computer Science, University of Georgia, USA) and in collaboration with industry and government R&D (Naval Research Laboratories, USA), and application partners (Medical College of Georgia, University of Georgia, USA), we have developed the METEOR workflow management model and system. METEOR's architecture includes design, monitor, workflow repository, and enactment services. METEOR has adopted semantics and Web services to face the new challenges of the semantic Web. The METEOR-S (METEOR for Semantic Web services) [1] project is being developed in collaboration with IBM TJ Watson re-search center. METEOR-S is focused on the usage of semantics for the complete lifecycle of semantic Web processes, which represent complex interactions between semantic Web services. METEOR-S project targets research on four important areas of the lifecycle of semantic Web processes [2], namely, annotation, discovery, composition, and orchestration. For each of the research stages in the lifecycle a framework, infrastructure or environment has been developed and implemented.

The enactment service ORBWork has been developed to address specific needs of organizations. ORBWork [3] is a CORBA [4] based system oriented to support mis-sion-critical enterprise applications requiring high scalability and robustness. It is fully distributed, scalable, and includes multilevel security mechanisms [5]. It supports interoperability standards such as JFLOW [6] and SWAP [7]. With recently added modules, it also includes an interface that allows for the realization of dynamic changes at the instance level, an exception-handling mechanism that is part of the adaptation module, and support for QoS-based adaptation and analysis. In this paper we will address the issues associated with these modules.

3. METEOR'S ARCHITECTURE

The adaptation of systems is a complex issue. This is even more so in distributed systems because of the existence of underlying infrastructures that are not frequently encountered in more traditional centralized systems. The adoption of adaptation strategies has a critical impact on early decisions in system development; it is both cost-effective and efficient to conduct adaptation analyses at the architecture level, before substantial resources have been committed to development (Bass, Clements, & Kazman, 1998). Therefore, the first step is to conduct an analysis in order to obtain a clear and accurate understanding of WfMSs' architecture.

We extend Worah and Sheth's (1997) classification of WfMSs' architecture with the addition of the schema layer. Therefore, we view workflow systems as having a four-tier layered architecture (Cardoso, Luo, Miller, Sheth, & Kochut, 2001): instance level, schema level, workflow level, and infrastructure level. Each level corresponds to a functional division of the WfMS, and each has a precise and specific mission (Fig. 1).

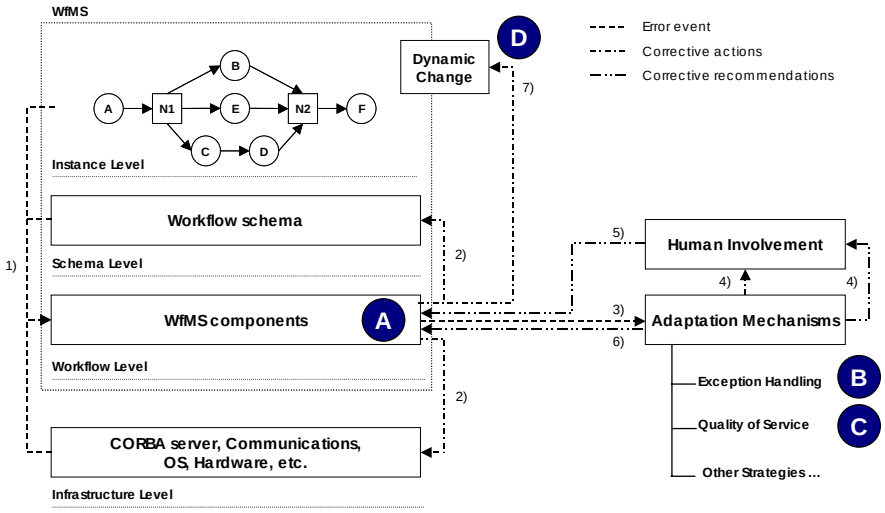


Fig. 1. Adaptation architecture for WfMSs

To better understand the purpose of each architectural level, we briefly describe them.

- Infrastructure Level. The infrastructure level includes all the elements that compose the underlying WfMS infrastructure. It includes CORBA servers, databases, operating systems, communication protocols, hardware, etc.
- Workflow level. At the workflow level, we find the modules that compose a workflow management system. The modules typically include the enactment engine, the monitor, and the repository.
- Schema Level. As the name suggests, the schema level includes workflow schema definitions. A workflow schema is a business process representation that can be interpreted by a WfMS. Workflow schemas are generally stored in a repository and are subsequently used by the WfMS.
- Instance Level. At the instance level, we find issues that are closely related to workflow instances or application executions. In this level, adaptation may occur when the design of a workflow schema does not anticipate a possible error related to the execution of workflow tasks.

4. ADAPTATION AND WORKFLOW MANAGEMENT SYSTEMS

As Charles Darwin noted, “it is not the strongest species that survive, or the most intelligent, but the one most responsive to change.” It is also desirable for a workflow management system to be adaptable. Adaptive mechanisms dynamically change active workflow instances. This permits WfMSs to be able to adapt themselves to a range of different business and organization settings, and also to a changing context (Han, Sheth, & Bussler, 1998). This requirement is a direct consequence of the highly changeable environment in which business processes normally operate.

Fig 1 presents our overall WfMS adaptation architecture. This architecture includes: the WfMS and the functional levels (the instance, schema, workflow, and infrastructure levels) at which specific types of events can occur; the detection and forwarding of events between levels (represented by 2); and event handling (represented by 1, 3, 4, 5, 6, 7).

A typical scenario that requires adaptation is described as follows. When an event is generated at any functional level, it is the responsibility of the level where the event occurred to restore its own consistency. Depending on the type of event, it may not be possible for the level to self-adapt. This is because additional knowledge may be needed to assess the event which is simply not available to the level. In such a case, the event is forwarded (1) to the workflow level. Upon receiving the report of an event, the workflow level will try to find a local solution to the problem. The workflow level is a good candidate for deriving a solution since it is the central point that coordinates the instance and schema levels. If a solution is found locally (described in section 4.1 and indicated in Fig. 1 with the letter ‘A’), then a set of adaptive actions are applied

(2, 7). On the other hand, if the workflow level cannot find a solution, the event is forwarded (3) to the adaptation module.

The adaptation module is a specialized service that incorporates knowledge and various algorithms to derive solutions for specific problems. When the adaptation module receives an event, it tries to derive a valid solution. Depending on its implementation, the adaptation module can rely on several distinct methods to obtain a solution. In section 4.2 we describe two adaptation methods – one to deal with workflow exceptions and the other one to manage QoS requirements (indicated in Fig. 1 with the letters 'B' and 'C', respectively). If a solution cannot be found or if only a partial solution is created, then this information is forwarded (4) to another module which waits for human involvement. During the procedure described, at any point when a valid solution is derived it is sent (5, 6) to the workflow level, which applies corrective actions. If corrective actions are necessary at the instance level, it is necessary to use the dynamic change (described in section 4.3 and indicated in Fig. 1 with the letter 'D'), interface (7) to guarantee that the changes are applied in a consistent manner.

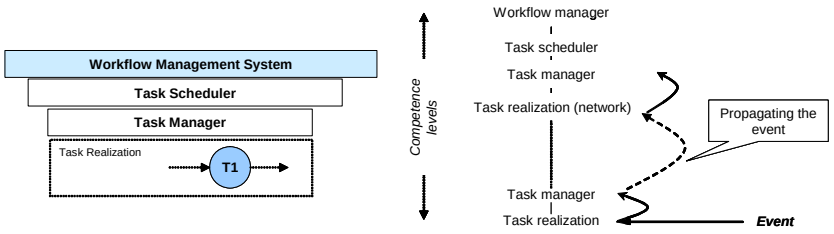
4.1 Triggering Adaptation

Adaptation strategies can be hard-coded into the workflow system. In this case, they are called built-in. The advantage of using such strategies is that the user does not have to set up or configure any mechanism in order for the workflow system to handle events. The disadvantage is that there is no flexibility for a user to customize the system.

In METEOR, events are represented by exceptions. When an event is identified in a component, an exception is generated. METEOR's system architecture is composed of several components that form the component hierarchy depicted in Fig 2.a; this includes the WfMS, the task scheduler, the task manager, and task realization.

Each component is said to be competent to handle an event if it has a handler associated with it. Having a handler for an event indicates that the component has the knowledge and techniques to derive a solution for the problem. For that reason, the component hierarchy shown in Fig 2.b induces a *competence hierarchy* (Kochut, Sheth, & Miller, 1999).

The strategy used to deal with events is called the *competence-driven exception handling strategy*. It works in the following way. In the case that an event occurs, it is first made available to the workflow component in which the flow of control resides. If this component is competent to handle the event, i.e. if it has at least one handler for the given event type, it deals with the event. On the other hand, if the component does not have the competence for the event, the event is passed on to the next higher level in the competence hierarchy.



a) b)

Fig. 2. METEOR model competence hierarchy

Since the events form a hierarchy, a given task may have more than one suitable handler. In this case, the most specific handler is used. The workflow runtime system has default handlers for all workflow-related events. In other words, the workflow manager is competent to handle any event, whether system or application-defined. The handlers invoke an adaptation mechanism whenever necessary and wherever possible. If an event is detected while a task manager is attempting to run a task realization (for example, a

computer program), the task manager itself may be competent to handle the event with one of its internal handlers. Such handlers typically involve retrying the task realization a number of times, before throwing the event to a higher-level component.

4.2 Adaptation Strategies

In this section we present two runtime time strategies which increase the adaptation of workflow systems: exception handling and Quality of Service (QoS) management. To illustrate the use of adaptation, we describe two adaptable systems that work in conjunction with the METEOR system. The first system deals with exceptions – a well-defined class of events that may occur during the realization of a workflow instance. The second system is responsible for adapting workflow instances when their Quality of Service (QoS) reaches threshold values. While the first system handles functional errors, the second one deals with operational errors.

4.2.1 Adaptation and Exception Handling

The need for modeling exceptions in workflow systems has been widely recognized (Casati, 1998). An exception refers to abnormal events not modeled by the underlying workflow management system, deviations between what we plan and what actually happens (Luo, 2000).

The architecture developed (Luo, 2000) and implemented in the METEOR system includes a sophisticated exception-handling mechanism. During a workflow schema execution, if an exception occurs is propagated to the case-based reasoning (CBR) exception handling module, the CBR system is used to derive an acceptable exception handler (Luo, Sheth, Miller, & Kochut, 1998). The system has the ability to adapt itself over time, based on knowledge acquired about past experiences that will help to solve new problems. As the CBR system collects more cases, the WfMS becomes more resistant, thus preventing unwanted states. This is because it has a larger set of knowledge to handle future exceptions.

Human involvement is needed when acceptable exception handlers cannot be automatically obtained. Solutions provided by a person will then be incorporated into the case repository. The effects of the exception-handler candidates on the workflow system and the applications are then evaluated. Thus, when an exception is handled, modifications to the workflow system or applications may be needed. The exception-resolution process is actually the population process of the CBR templates. The exception-resolution procedure performs the following tasks (Luo, 2000):

- The *coordination mode* of exception handling is determined according to the type of process interactions among business processes.
- The *contacting party* as well as an interaction point is determined. A contacting party is the entity that is responsible for handling exceptions in the processes of its organization. An interaction point is where the interactions can take place.
- The *compensation scheme* is found, if necessary. The nature of the processes will affect the compensation schemes. When a compensation schema does not exist, human involvement is allowed to determine the compensation schema.
- The *rework scheme* is found, if necessary. A rework scheme is a plan for the processes to make progress from the failure point.

The retrieval procedure of similar previous cases is based on a similarity measure that takes into account semantic and structural similarities and differences among the cases. A similarity measure is achieved by obtaining the following:

- *Exception similarity*. Exception similarity is based on the is-a relationship in the exception hierarchy in METEOR model 3 (Kochut, 1999).
- *Workflow similarity*. This is a workflow structural similarity, such as AND, OR building-block similarity.
- *Context similarity*. This is obtained by computing the nearest neighborhood function of the quantified degrees of semantic similarities over workflow application data. To accomplish this, a concept tree is built, and the distances between concepts are stored in the case repository.

We use a pattern-guided case adaptation scheme. There are four steps in the adaptation process in this pattern-guided adaptation scheme.

- *Classifying the exception pattern*. At this step, the exception pattern is identified. If it is a new pattern, it is added to the exception pattern repository.
- *Searching the handling pattern.* Once the exception pattern is determined, a search is conducted for the handling pattern. At this step, the exception handling coordination mode is determined. The contacting party, as well as the interaction point, is also determined by analyzing the interactions among business processes.
- *Selecting a handler pattern.* A handler pattern is selected, based on the search resulting from step 2. The compensation scheme as well as the rework scheme is determined.
- *Initializing the handler*. The CBR handling template is populated. A new case is created which results from the adaptation of an existing one.

4.2.2 Adaptation and Workflow Quality of Service

Workflow QoS represents the quantitative and qualitative characteristics of a workflow application necessary to achieve a set of initial requirements (Cardoso, Miller, Sheth, Arnold, & Kochut, 2004). It addresses the non-functional issues of workflows. Quantitative characteristics can be evaluated in terms of concrete measures, such as workflow execution time and cost, while qualitative characteristics specify the expected services offered by the system, such as security and fault-tolerance mechanisms.

In METEOR, workflows follow a QoS model (Cardoso et al., 2004) composed of three dimensions: time, cost, and reliability. The tasks and transitions of a workflow are initialized with QoS estimates. To determine useful estimates, a combination of a priori estimates from designers and estimates computed from prior executions are used, with the historical data playing a larger role as more data is collected. Once QoS estimates are determined, it is then possible to compute a workflow QoS. The computation can be achieved using a mathematical (Cardoso et al., 2004) or a simulation approach (Cardoso & Sheth, 2003). The selection of one of the methods is based on a tradeoff between time and accuracy of results.

At runtime, the quality of service of workflow instance is monitored. The monitoring tools seek to identify the QoS status of instances and the possible occurrence of deviations from the desired metrics. When QoS deviations are identified, adaptation mechanisms are used to derive and apply reparative actions to the faulty instances.

Let us consider the following example. Several workflow instances are running correctly and the quality of service specifications are being followed when a task starts to exhibit poor time QoS metrics. As a consequence, the workflow QoS specifications of time are no longer satisfied. The adaptation module identifies this problem and uses a problem-solving technique to derive a solution. In this case, the adaptive system infers that the faulty task needs to be replaced by an equivalent task. This replacement can be accomplished by applying dynamic changes to the workflow instances, either manually or automatically (Cardoso et al., 2001).

Automatically finding a task to replace a faulty one is challenging. Therefore, this step may in some cases require human intervention. Nevertheless, to minimize human involvement, sophisticated discovery mechanisms to search for tasks can be implemented. In some cases, a discovery mechanism can search for and find a task without human intervention. One approach to increase the degree of automatism is to describe tasks and their operational metrics semantically (Cardoso & Sheth, 2003). This description allows the discovery process to find tasks not only based on their functionality, but also based on their operational metrics, i.e. based on their QoS. Once a task with the required functionality and QoS is found it can replace the faulty task in a workflow.

4.3 Adaptation and Dynamic Changes

The two adaptation methods described in the previous sections cannot successfully accomplish their objectives without having the support of a dynamic change layer. Adaptive methods rely on a dynamic change layer to adapt running instances.

Traditional workflow systems are adequate to support business processes which are static (Reichert & Dadam, 1998), with no need for the support dynamic changes at runtime. But, recently there has been an

increasing demand for developing systems with dynamic capabilities (Horn & Jablonski, 1998; Weske, Vossen, & Medeiros, 1996), with a special emphasis on dynamic changes at the workflow instance level.

As workflow processes are instantiated, changes in the environment or generated by previous activities may invalidate the current workflow instances, requiring reparative actions (Berry & Myers, 1998). Long-running applications which are heterogeneous, autonomous, distributed may require support for dynamic reconfiguration when machines fail, services are moved or withdrawn, and user requirements change. In such environments, it is essential that the structure of applications be modified to reflect these changes (Shrivastava & Wheater, 1998).

We classify the different types of change into two main categories: primitive change and composite change. Primitive change can be further divided into immediate change and incremental change. Immediate changes can be introduced in one step. Incremental changes, on the other hand, deal with situations where we cannot apply the changes to a particular instance in a one-step procedure. For example, if a set of instances is waiting for the completion of a task t, and if we dynamically change t specifications, the waiting instances may enter an inconsistent state, since the information that they have relative to the previous task t no longer reflects the current state of the system.

In our work, most of the primitive changes implemented are incremental changes (Tabel 1). Composite changes are composed of a sequence of primitive changes that describe an elaborate process of definition change (e.g., adding a task between two existing tasks is the result of applying a sequence of primitive changes).

Table 1. Dynamic change classifications

Dynamic Change	Change Type	Implemented
AND to OR Join Change	Incremental	Yes
OR to AND Join Change	Incremental	Yes
Split Change	Incremental	Yes
Addition of an AND Transition	Incremental	Yes
Addition of an OR Transition	Incremental	Yes
Deletion of a Transition	Incremental	Yes
Data Object Transfer Addition	Incremental	Yes
Data Object Transfer Deletion	Incremental	Yes
Parameter Mapping Change	Incremental	Yes
Parameter Type Change	Incremental	No
Task Type Change	Incremental	No
Task Invocation Change	Composite change	No
Insertion of a Task	Composite change	Yes
Deletion of a Task	Composite change	No

In the ORBWork system, we have implemented a layer that permits the consistent realization of the dynamic change of instances (Chen, 2000). The implemented module guarantees that all consistency constraints which have been ensured prior to a dynamic change are also ensured after the workflow instances have been modified (Reichert & Dadam, 1998). The dynamic change interface was built on top of the CORBA ORB infrastructure, using IIOP as the underlying communication protocol. Additional functions have been added to the IDL interface of the CORBA object responsible for managing tasks.

5. RELATED WORK

To support dynamic workflow adaptations, Greiner et al. (Greiner et al., 2004) have developed a rule-based approach for the detection of logical failure events. Their approach has been implemented within the WfMS ADEPTflex and workflow adaptation system AdaptFlow. Their systems use a rule-based approach for the detection of logical failure events (exceptions). In our system we use a CBR approach. We believe that a CBR system is more adaptive since as the CBR system collects more cases, the WfMS becomes more resistant since it has a larger set of knowledge to handle future exceptions. Moreover, their approach only

deals with logical failure events, while we also address the problem of adapting instances to maintain a specified level of Quality of Service.

Adams et al. (Adams, Edmond, & Hofstede, 2003) extend the applicability of Activity Theory to the implementation of more flexible WfMSs. Activity theory offers a number of interesting solutions for workflow adaptability, flexibility, evolution and exception handling. Their research describes the activity theory principles that should be implemented when developing WfMSs. Unfortunately, no implementation has been done, and their work does not address QoS issues.

Siebert (Siebert, 1999) proposes an integrated approach for adaptive workflow support. His work describes extensions to the SWATS architecture in order to enable the executeon of unstructured process and illustrates how an adaptive support layer can be integrated to extend existing workflow management systems. The adaptive layers presented do not address directly the problem of handling exceptions nor QoS monitoring and adaptation.

6. CONCLUSIONS

The new requirements of modern systems in our highly technological society demand that critical systems be adaptable. Our work focuses on the adaptation of workflow management systems (WfMSs).

In this paper we have presented a set of comprehensive techniques to be used in the development of WfMSs to increase their level of adaptation. To develop a successful solution, the first step is to develop a conceptual architecture for workflow systems which will provide a fundamental framework for developing an adaptable WfMS architecture. We have defined an adaptable architecture that functionally divides WfMSs into four levels: instance level, schema level, workflow level, and infrastructure level. Thereafter, we start by discussing how the need for adaptation can be triggered. We not only target adaptation from a functional perspective, but also from an operational perspective. It is important to understand that adaptation does not restrict its span to functional errors, i.e. is the workflow system or are the tasks working or not? Adaptation also addresses operational issues, such as the QoS management of workflow instances. To illustrate how functional and operational deviations can be handled, two adaptive modules are described. The first one deals with exceptions and the second one targets the QoS of workflows. Finally, we explain the importance and discuss the development of a dynamic change layer to carry out adaptation strategies.

Although the majority of the ideas presented were implemented in METEOR system, the concepts and ideas are independent of the WfMS chosen. Therefore, it is, in principle, possible to add the notions presented in this paper to most of the workflow management systems available today.

ACKNOWLEDGEMENT

We would like to acknowledge Krys Kochut's work on the specifications of METEOR Model 3 and the implementation of the METEOR competency hierarchy to handle exceptions. This research was funded, in part, by the Naval Research Laboratory sponsored "Workflow Management for Advanced DoD Applications" and "Extending METEOR with Workflow Reuse, Adaptation, and Collaboration" projects.

REFERENCES

Adams, M., Edmond, D., & Hofstede, A. t. (2003). *The application of Activity Theory to dynamic workflow adaptation issues.* Paper presented at the 7th Pacific Asia Conference on Information Systems (PACIS 2003), Adelaide, South Australia.

Bass, L., Clements, P., & Kazman, R. (1998). *Software Architecture in Practice*: Addison Wesley.

Berry, P. M., & Myers, K. L. (1998). *Adaptive Process Management: An AI Perspective.* Paper presented at the ACM Conference on Computer Supported Cooperative Work, Seattle, Washington.

Cardoso, J., Luo, Z., Miller, J., Sheth, A., & Kochut, K. (2001, March 16-17). *Survivability Architecture for Workflow Management Systems.* Paper presented at the Proceedings of the 39th Annual ACM Southeast Conference, Athens, GA.

Cardoso, J., Miller, J., Sheth, A., Arnold, J., & Kochut, K. (2004). Modeling Quality of Service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web Journal, 1*(3), 281-308.

Cardoso, J., & Sheth, A. (2003). Semantic e-Workflow Composition. *Journal of Intelligent Information Systems (JIIS). 21*(3), 191-225.

Casati, F. (1998). *A Discussion on the Approaches to Handling Exceptions in Workflows.* Paper presented at the Proceedings of 1998 Computer-Supported Cooperative Work (CSCW 1998), Towards Adaptive Workflow Systems Workshop, Seattle, WA.

Chen, Y. (2000). *Design and Implementation of Dynamic Process Definition Modifications in OrbWork Enactment System.* Unpublished M.Sc. Thesis, University of Georgia, Athens, GA.

Greiner, U., Ramsch, J., Heller, B., Löffler, M., Müller, R., & Rahm, E. (2004). *Adaptive Guideline-based Treatment Workflows with AdaptFlow.* Paper presented at the Proc. of Symposium on Computerized Guidelines and Protocols (CGP 2004), Prague.

Han, Y., Sheth, A. P., & Bussler, C. (1998, November 1998). *A Taxonomy of Adaptive Workflow Management.* Paper presented at the Workshop of the ACM 1998 Conference on Computer Supported Cooperative Work, Seattle, WA.

Horn, S., & Jablonski, S. (1998, November 1998). *An Approach to Dynamic Instance Adaption in Workflow Management Applications. Workshop on Adaptive Workflow Systems.* Paper presented at the Conference on Computer Supported Cooperative Work (CSCW), Seattle, WA, USA.

Kochut, K. J. (1999). *METEOR Model version 3*. Athens, GA: Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia.

Kochut, K. J., Sheth, A. P., & Miller, J. A. (1999). *ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR*. Athens, GA: Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia.

Luo, Z. (2000). *Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes.* Unpublished Ph.D. Dissertation, University of Georgia, Athens, GA.

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

Reichert, M., & Dadam, P. (1998). ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems - Special Issue on Workflow Managament, 10*(2), 93-129.

Shrivastava, S. K., & Wheater, S. M. (1998). *Architectural Support for Dynamic Reconfiguration of Distributed Workflow Applications.* Paper presented at the IEEE Proceedings Software Engineering.

Siebert, R. (1999). An Open Architecture for Adaptive Workflow Management Systems. *Transactions of the SDPS: Journal of Integrated Design and Process Science, 3*(3), 29-24.

Weske, M., Vossen, G., & Medeiros, C. B. (1996). *Scientific Workflow Management: WASA Architecture and Applications.* Paper presented at the Fachbericht Angewandte Mathematik und Informatik 03/96-I, Universitat Munster.

Worah, D., & Sheth, A. P. (1997). Transactions in Transactional Workflows. In S. Jajodia & L. Kerschberg (Eds.), *Advanced Transaction Models and Architectures* (pp. 3-34): Kluwer Kluwer Academic Publishers.