

SEMANTICFLOW: A SYSTEM FOR FLEXIBLE WORKFLOW EXECUTION

Tatiana Almeida Souza Coelho Vieira
Marco Antonio Casanova
Department of Informatics
Pontifical Catholic University of Rio de Janeiro, Brazil
Rua Marques de S. Vicente, 225 - Rio de Janeiro, RJ - Brazil - CEP 22.453-900
{tati,casanova}@inf.puc-rio.br

Keywords: Workflow, distribution, cooperation, synchronization, presuppositions, execution alternatives.

Abstract: Workflow management systems received considerable attention lately, motivated by their wide spectrum of applications. Requirements for workflow management systems comprise a long list, among which one may highlight distributed execution, cooperation and coordination, and synchronization, that model the way user communities work cooperatively to perform a given task. This doctoral dissertation addresses a complementary family of requirements, collectively called *flexible execution*. Briefly, workflow management systems usually interpret a workflow definition rigidly. However, there are real life situations where users should be allowed to deviate from the prescribed sequence of actions for various reasons, including the unavailability of required resources. To cope with situations such as this, mechanisms to flexibilize workflow execution are proposed, that allow execution to proceed in the presence of incomplete information, by adopting presuppositions, and in the presence of negative information, by suggesting execution alternatives. An implementation of the mechanisms, that extends known Web service frameworks, is also outlined.

1 INTRODUCTION

Workflow management systems received considerable attention lately, motivated by their wide spectrum of applications. Standardization efforts are under way in the context of consortiums, such as W3C, OASIS and WfMC. Among other contributions, these efforts resulted in new workflow definition languages and coordination protocols. In particular, WfMC was created in 1993 by several companies with the purpose of standardizing the implementation of workflow management systems (Hollingsworth, 1995).

Requirements for workflow management systems comprise a long list, among which we may highlight distributed execution, cooperation and coordination, and synchronization, that model the way user communities work cooperatively to perform a given task (Georgakopoulos et al., 1995; Alonso and Schek, 1996; Alonso et al., 1997).

This doctoral dissertation addresses a complementary family of requirements, that we collectively call *flexible execution*. Briefly, workflow management systems usually interpret a workflow definition rigidly. However, there are real life situations where users should be allowed to deviate from the prescribed se-

quence of actions for various reasons, including unavailability of the required resources. The first goal of this doctoral dissertation therefore is to flexibilize workflow execution by introducing:

- *a mechanism to handle presuppositions* made in the presence of incomplete information that allow workflow execution to proceed;
- *a mechanism for choosing alternatives* to subworkflows, resources and users, thereby allowing workflow execution to proceed when the predefined subworkflow or resource, or the pre-assigned user is unavailable (i.e., in the presence of negative information).

Such mechanisms will use additional semantic information about the workflow definitions, resources and users involved to:

- suggest presuppositions when the required information is unavailable;
- propose alternative subworkflows, resources and users when the predefined subworkflow or resource, or the pre-assigned user is unavailable;
- assess the presuppositions made and the alternatives taken when workflow execution terminates.

The second goal of this doctoral dissertation is to propose an implementation of the mechanisms, in the context of Web services, that extends known service frameworks (Cabrera et al., 2002a). In (Papazoglou, 2003) the author presents an extended discussion about how to take advantage of the Web Services technology in the business transaction world. This topic is briefly discussed towards the end of the paper.

The published literature and the software industry provide many examples of workflow technology in general. In (Du and Elmagarmid, 1997; Mohan, 1997), the authors present the current capabilities of workflow products and point out some research issues in the area. Our research focus is closely related to two of these issues - distributed workflow execution and management of dynamic workflows.

Despite the large number of workflow systems implemented, the flexible execution requirement has been scarcely considered.

The idea behind the flexibilization mechanisms proposed in this doctoral dissertation is very similar to the query relaxation strategy used in the CoBase system (Chu et al., 1994) and in the CoSent system (Chu and Mao, 2000).

This paper is organized as follows. Section 2 presents some of the basic workflow concepts and outlines the basic distributed workflow execution model. Section 3 describes the flexibilization mechanisms we propose. Section 4 outlines an implementation of the mechanisms in a context of Web services. Finally, Section 5 contains the conclusions and summarizes the next steps of the doctoral dissertation.

2 BASIC WORKFLOW CONCEPTS AND EXECUTION MODEL

This section presents some of the basic workflow concepts and outlines the workflow execution model we base our flexibilization mechanisms.

2.1 Basic Concepts

A workflow definition typically contains a rigid indication of the tasks to be executed, a description of the resources necessary to perform the tasks and a user profile that defines who (or what) is suitable to perform the tasks. In particular, the user profile may be simply a description of the user or device capabilities required to perform the tasks.

In more detail, a workflow definition involves three familiar concepts:

- *tasks*, which are the smallest unit of execution of workflows. Tasks, and their execution order,

are statically described in the workflow definition. Each task execution results in a task instance, which keeps all the required state information. The definition of tasks and (sub)workflows is static, but the state information is not;

- *resources*, which are used during task execution. The list of resources a task uses is part of the definition of the task;
- *users*, which are responsible for the execution of tasks. We use the generic term “user” to refer to any (human) user, group of (human) users, hardware device or software component that is responsible for executing tasks.

Tasks can be *replicable* or *non-replicable*. A replicable task can be simultaneously given to many users for execution, whereas a non-replicable task cannot. In addition, replicable tasks can be *inclusive* or *exclusive*. The replication protocol proceed with workflow execution only when it receives the results from all instances of an inclusive replicable task and verifies that they are consistent with each other. By contrast, the replication protocol proceeds with workflow execution when it receives the result of the first instance of an exclusive replicable task that terminates; when this occurs, the protocol cancels all other instances of the exclusive tasks.

2.2 Basic Workflow Execution Model

To become eligible to participate in the execution of a workflow instance, a user must register himself with the system. After registering, the user becomes *available* to the system. The registration protocol basically represents an agreement between the system and the user that describes the execution capabilities of the user.

When a user U starts a workflow instance W , he becomes the *coordinator* of W . User U may delegate a nested subworkflow W' of W to another available user U' , perhaps with permission to redistribute subworkflows. Then, U' becomes the coordinator of W' . We call U the *manager* of U' and U' a *subordinate* of U . Hence, we effectively have a *coordination hierarchy* for W , rooted at U , which is also called the *central coordinator* of the hierarchy.

The system assigns an identifier to each workflow or subworkflow W'' and creates a common data space B'' for W'' , called here a *blackboard*. Among other data, B'' will keep the values of all variables that W'' uses, the identifier of W'' and the address of the manager of the coordinator U'' of W'' . Hence, U'' has direct access to the blackboard of its manager, and indirect access to the blackboard of all its ancestor managers.

Finally, the system treats W , with its nested subworkflow instances, as a *workflow transaction*, in

the sense of (Worah and Sheth, 1997; Worah and Sheth, 1996).

In summary, our distributed workflow execution model can be viewed as an hierarchy of three basic components: coordinators; workflow instances; and blackboards. This execution model dilutes the burden of controlling the distributed execution of a workflow instance among a hierarchy of coordinators and avoids the danger of transforming the central coordinator into a hotspot.

3 FLEXIBLE EXECUTION MODEL

Usually, a workflow management system executes a workflow exactly as defined, which implies that the system will wait until the required resources or users are known (to the system) to be available, if at all. This rigid execution may hinder performance or may even render workflow execution useless.

This section describes two mechanisms that allow workflow execution to proceed in the presence of incomplete information, by adopting presuppositions, and in the presence of negative information, by suggesting execution alternatives.

3.1 Handling Presuppositions

The mechanism to handle presuppositions is invoked whenever the required information is unavailable. This mechanism is based on two points:

1. sets of presuppositions for variables, that is, sets of values, defined by rules, to be assumed for the variables;
2. a set of consistence criteria that define when an execution based on presuppositions is valid.

As an example, consider Table 1, where A, B, C e D indicate the environmental impact of each of the procedures for cleaning beaches when an oil spill occurs (assuming that there are 5 types of oil). A indicates the smallest environmental impact, B some impact, C a significant impact and D the greatest impact.

Suppose that a user needs to select a cleaning procedure for a beach affected by an oil spill, but he has no information about the oil type. Also assume that the presupposition mechanism has a rule saying that, if the oil type is unknown, then type II should be assumed. Then, the user does not have to wait until the oil type is known; he can invoke the presupposition mechanisms, assume that the oil type is II and proceed with the execution.

The mechanism to handle presuppositions may be invoked in two distinct points of the execution. First,

when delegating a subworkflow instance W' of a workflow instance W to another user U' , the coordinator U of W may find out that the value of a variable V , required to test the pre-condition C' of W' , is missing. In this case, U executes the presupposition mechanism to find out a value p for V , tests C' with p , and delegates W' to U' , if C' succeeds when V receives p as value. Second, when is executing W , he may also miss the value of some variable T . Then, U may again invoke the presupposition mechanism to find out a value for T .

When a subworkflow instance terminates, a consistence check is necessary to analyze the results produced. Suppose that the consistency check depends on a set S of variables. Then, this check can be performed by the first coordinator in the coordination hierarchy, from the leaves to the root, that has access to the values of all variables in S .

3.2 Choosing Alternatives

The mechanism for choosing alternatives allows workflow execution to proceed even when the predefined subworkflow or resource, or the pre-assigned user is unavailable (i.e., in the presence of negative information).

There are basically three alternatives:

- find a semantically equivalent subworkflow such that the required resources and the assigned user are available;
- find semantically equivalent resources that are available;
- find a user, with equivalent capacities, that is available.

It is worth noting that these three choices are mutually dependent. For instance, if an alternative subworkflow is considered, one must verify if the required resources and users are available. Likewise, if an alternative user is considered, one must check if he can really execute that subworkflow and if the resources are available.

The mechanism for choosing alternatives uses additional semantic information about the workflow definitions, resources and users involved, as follows:

- a workflow ontology that captures semantic relationships between workflows;
- a resource ontology that captures semantic relationships between resources;
- a user ontology that captures capacity equivalences between users.

Moreover, typically more than one alternative is possible. Hence, the mechanism for choosing alternatives must be equipped with a cost model that is invoked to select the best alternative, or at least to guide

	Type I	Type II	Type III	Type IV	Type V
Natural Degradation	A	A	A	A	A
Manual Cleaning			B	B	B
Use of Absorbents		B	A	A	A
Vacuum Cleaning		A	A	A	A
Cold, low pressure cleaning		A	A	B	B
Cold, high pressure cleaning		B	B	B	B
Hot, low pressure cleaning			C	C	C
Vapor cleaning			D	D	D

Table 1: Environmental impact of beach cleaning procedures according to oil type.

some heuristics that will select a reasonable alternative.

Consider Table 1 already used in 3.1. When the accident is characterized by an oil spill that reached a beach, several cleaning procedures can be used. If the oil is of type II, the best procedures are “natural degradation” (i.e., do nothing), “vacuum cleaning” and “cold, low pressure cleaning”. If it is impossible to execute these procedures because a predefined resource is unavailable, then the “use of absorbents” and the “cold, high pressure cleaning” procedures should be used, according to the environmental impact qualification (A, B, etc.) of each available procedure.

3.3 Using the Flexibilization Mechanisms

When a user registers to participate in a particular workflow instance, he must choose the execution protocol that will dictate his behavior in the system. There are two possible protocols: the waiting protocol and the protocol to handle presuppositions.

According to the waiting protocol, the user will wait until the values of all variables he requires to proceed execution become available. By contrast, according to the protocol to handle presuppositions, the user may request a (presupposed) value for a variable, if none is available, to proceed with the workflow execution.

As already discussed, during the delegation process, a coordinator may discover that some of the variables she/he needs to delegate a subworkflow have missing values. If he registered with the system for the protocol to handle presuppositions, he can invoke it to generate a (presupposed) value for a variable, before delegating the subworkflow to the destination user.

A coordinator may also detect that a subworkflow cannot be delegated to a user because of negative information. In this case, he may invoke the mechanism for choosing alternatives to request an alternative resource or an alternative subworkflow. He may also

request an alternative user and maintain the original subworkflow.

Figure 1 presents a simplified state diagram of a coordinator. When the workflow instance starts, the coordinator enters the workflow pre-compiling phase. If necessary, the mechanism to handle presuppositions and the mechanism for choosing alternatives may be invoked. The next step is to delegate subworkflows to other users, adding the required information on the blackboard. After this step, the coordinator waits until he receives all results from their subordinated coordinators. When this occurs, the coordinator checks the consistence of the results, if they depend only on values stored on his blackboard. For each variable, its final value, computed locally, is propagated to the blackboard that stores the final value of the variable.

The termination protocol, executed by each coordinator, is similar to the two-phase commit protocol. However, because of the proposed flexibilization mechanisms, it must include a consistency check to verify the consistency of the overall transaction.

Figure 2 presents a simplified state diagram of a subworkflow instance, emphasizing the transitions that cover termination.

The initial state of a subworkflow instance is the “active” state. The instance remains in this state until execution begins. If the execution begins with a presupposition, the subworkflow goes to the “conditionally executing” state. If the execution begins without presuppositions, the new state is the “executing” state. From both states, the subworkflow instance can be cancelled or it can be ready to terminate with success. However, the final state of the workflow instance can only be reached after the execution of the termination protocol. Therefore, before the final state, there is always an intermediary state, where each subworkflow instance stays until it receives the termination message.

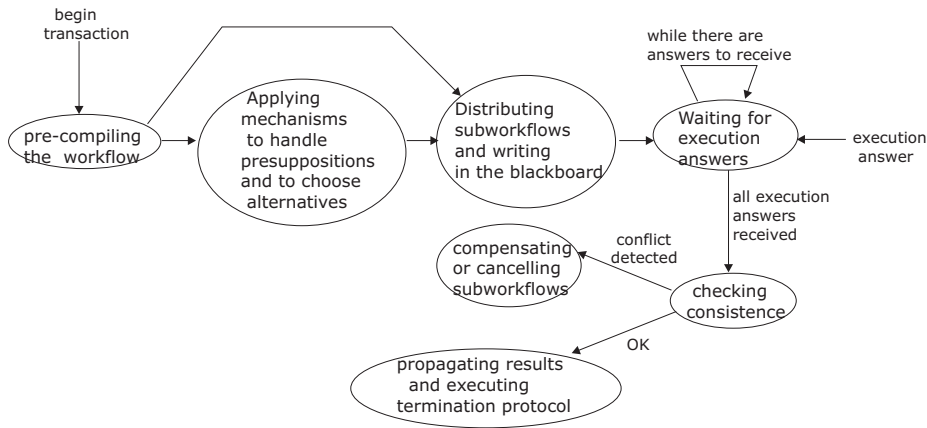


Figure 1: Simplified state diagram of a coordinator.

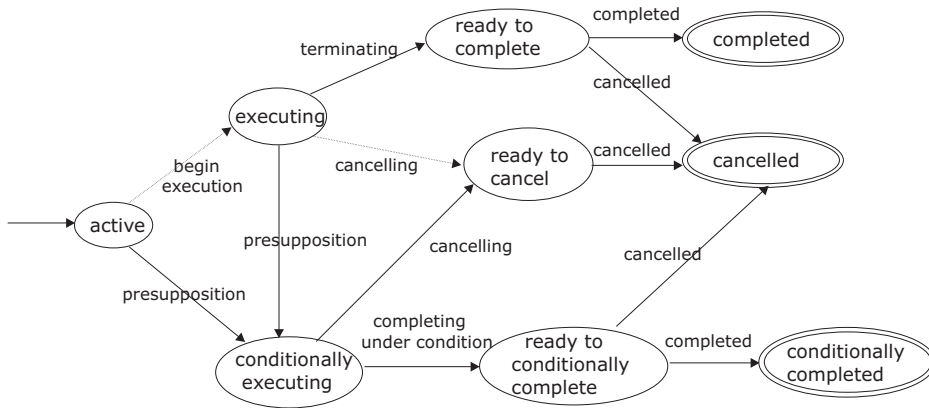


Figure 2: Simplified state diagram of a subworkflow.

4 IMPLEMENTATION IN THE CONTEXT OF WEB SERVICES

We outline in this section an implementation of the flexibilization mechanisms in a context where each workflow defines a composition of Web services. We refer to the system to be implemented in this doctoral dissertation as the *SemanticFlow* system.

We will adopt standard technology as much as possible. In particular, we will use OWL-S (formerly DAML-S) (Coalition, 2003) to describe the workflow, resource and user ontologies, and UDDI (Accenture et al., 2000) as the Web services discovery technology. A rule language, such as TRIPLE (Sintek and Decker, 2002), is also required to express presuppositions and consistency criteria.

OWL-S is sufficiently flexible to allow new properties to be added to service classes that capture details of our distributed computation model or our flexibilization mechanisms. For example, a boolean property may be added to indicate if a given service represents

a replicable task, or not.

The flexibilization mechanisms of *SemanticFlow* will be modelled as protocols, extending the *WS-Transaction* and *WS-Coordination* frameworks (Cabrera et al., 2002b). Figure 3 presents an schematic overview of how *WS-Coordination* and *WS-Transaction* work (Freund and Storey, 2002). Very briefly, an application exchanges context information with the Web services it invokes, including information about the transaction the invoked services is part of. The coordination modules exchange messages to coordinate the transaction and select the protocols to be used.

Figure 4 sketches how a workflow instance will be executed, assuming that there is just one coordinator and one blackboard, for simplicity. After a workflow instance starts, any Web service it invokes must register with the system, indicating which execution or flexibilization protocols it intends to use. The service may subsequently access the blackboard, and use any of the flexibilization protocols it registered for.

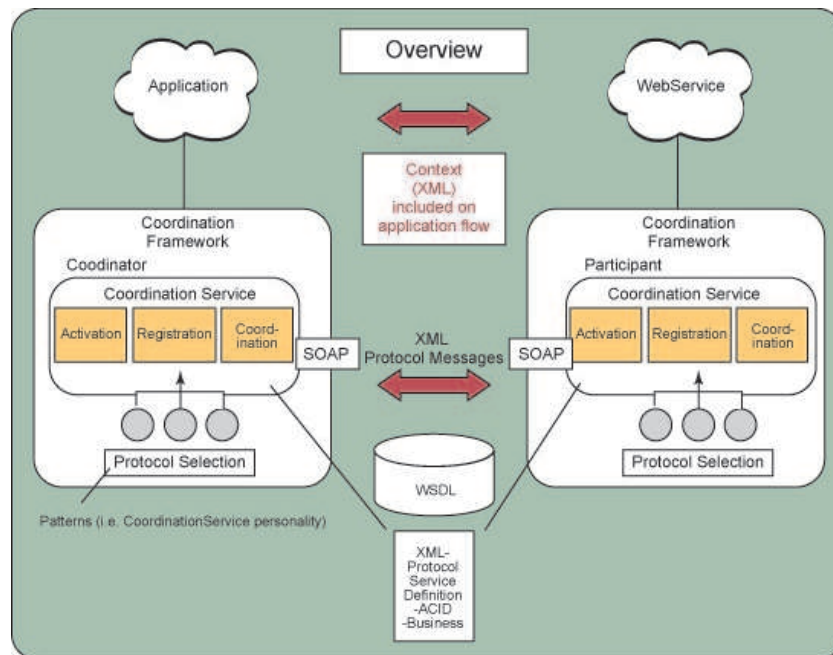


Figure 3: Schematic overview of WS-Coordination and WS-Transaction (Freund and Storey, 2002).

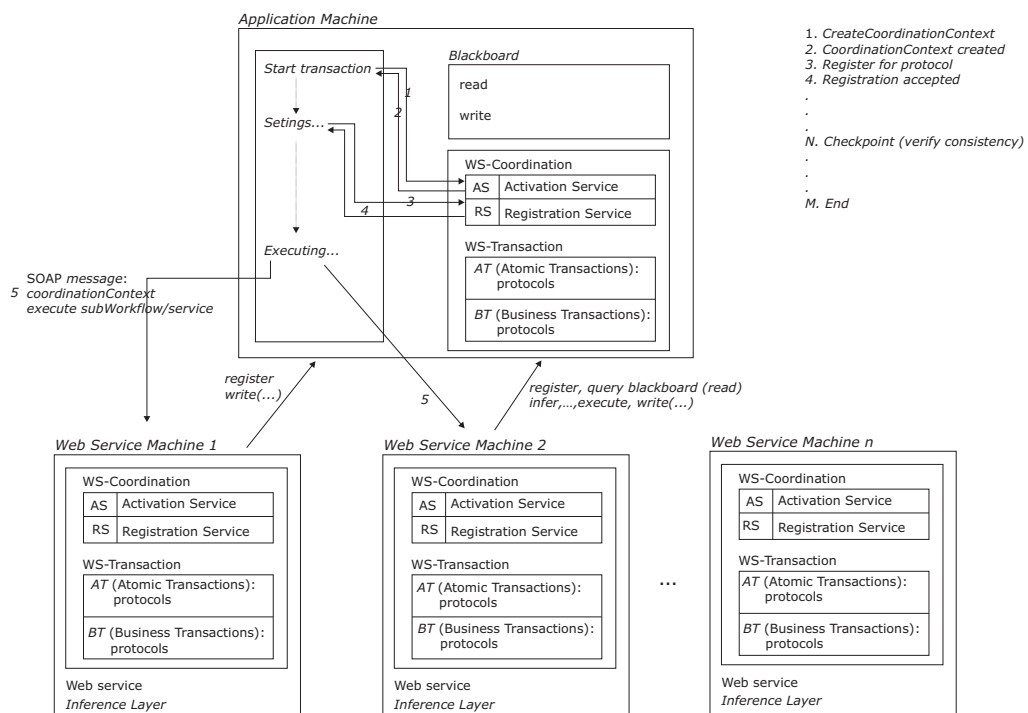


Figure 4: Execution of a workflow instance in the SemanticFlow system.

5 CONCLUSION

We proposed in this doctoral dissertation two mechanisms to flexibilize the execution of workflow instances: a mechanism to handle presuppositions that allows workflow execution to proceed in the presence of incomplete information, and a mechanism to choose alternative subworkflows, resources and users, in the presence of negative information.

These two mechanisms use additional semantic information about the workflow definitions, resources and users involved. This information can be very useful in many situations, specially when the execution must be terminated in a short period of time.

The idea is to validate the proposed mechanisms by implementing them using Web services technologies. Protocols and frameworks for Web Services must be extended and used to guarantee the flexibilization of the execution allowed by the proposed protocols.

The development of this doctoral dissertation follows five steps:

1. Bibliographical research: do research, aiming to investigate new workflow systems and solutions that are becoming available;
2. System specification: to formally specify the system and each one of the protocols, writing a consistent documentation that will guide the implementation;
3. Implementation investigation: to study strategies to implement the system, based on the semantic Web concept, and considering the new technologies involved;
4. Implementation: to implement the *SemanticFlow* system, following the specification, and giving priority to the flexibilization mechanisms proposed;
5. Benchmark: to evaluate how the proposed mechanisms behave for a meaningful set of workflow definitions.

We are carrying out the first three steps in parallel. The technologies to be used for implementing the mechanisms in the context of Web services are being investigated and, so far, we favor using the WS-Transaction framework (Cabrera et al., 2002a).

6 ACKNOWLEDGMENT

This work was partially supported by CNPq under grants 140600/01-9 and 55.2040/02-9.

REFERENCES

- Accenture, Inc., A., Inc., C. O., Limited, F., Company, H.-P., i2 Technologies Inc., Corporation, I., Corporation, I. B. M., Corporation, M., Corporation, O., AG, S., Inc., S. M., and Inc., V. (2000). UDDI Technical White Paper. <http://www.uddi.org>.
- Alonso, G., Agrawal, D., Abbadi, A. E., and Mohan, C. (1997). Functionality and Limitations of Current Workflow Management Systems. *IEEE Expert*, 2(5).
- Alonso, G. and Schek, H.-J. (1996). Research Issues in Large Workflow Management Systems. Technical Report 1996PA-as96-nsfws, Institute for Information Systems, Switzerland.
- Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T., and Thatte, S. (2002a). Web Services Transaction (WS-Transaction). <http://www.ibm.com/developerworks/library/ws-transpec/>. IBM, DeveloperWorks.
- Cabrera, F., Copeland, G., Freund, T., Klein, J., Langworthy, D., Orchard, D., Shewchuk, J., and Storey, T. (2002b). Web Services Coordination (WS-Coordination). <http://www.ibm.com/developerworks/library/ws-coor/>. IBM, DeveloperWorks.
- Chu, W. W., Chen, Q., and Merzbacher, M. (1994). *Studies in Logic and Computation: Nonstandard Queries and Nonstandard Answers*, volume 3, chapter CoBase: a Cooperative Database System, pages 41-72. Oxford University Press, New York. Edited by R. Demolombe and T. Imielinski.
- Chu, W. W. and Mao, W. (2000). CoSent: a Cooperative Sentinel for Intelligent Information Systems.
- Coalition, T. D. S. (2003). OWL-S 1.0 Release. <http://www.daml.org/services/owl-s/1.0/>.
- Du, W. and Elmagarmid, A. (1997). Workflow Management: State of the Art vs. State of the Products. HP Labs Technical Report HPL-97-90, HP Labs.
- Freund, T. and Storey, T. (2002). Transactions in the World of Web Services, Part 1: an Overview of WS-Transaction and WS-Coordination. <http://www-106.ibm.com/developerworks/library/ws-wstx1/>. IBM, DeveloperWorks.
- Georgakopoulos, D., Hornick, M. F., and Sheth, A. P. (1995). An Overview of Workflow Management: from Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119-153.
- Hollingsworth, D. (1995). The Workflow Reference Model. The Workflow Management Coalition Specification TC00-1003, Workflow Management Coalition, Hampshire, UK.
- Mohan, C. (1997). Recent Trends in Workflow Management Products, Standards and Research. In *Proceedings of the NATO Advanced Study Institute (ASI) on Workflow Management Systems and Interoperability*, Istanbul. NATO ASI Series, Series F: Computer

and Systems Sciences, Volume 164, Springer Verlag, 1998.

- Papazoglou, M. P. (2003). Web Services and Business Transactions. In *World Wide Web: Internet and Web Information Systems*, volume 6, pages 49–91, Netherlands. Kluwer Academic Publishers.
- Sintek, M. and Decker, S. (2002). TRIPLE—A Query, Inference, and Transformation Language for the Semantic Web. In *Proceedings of the International Semantic Web Conference (ISWC)*, Sardinia.
- Worah, D. and Sheth, A. (1996). What do Advanced Transaction Models Have to Offer to Workflows? In *Proc. of the International Workshop on Advanced Transaction Models and Architectures (ATMA)*, Goa, India.
- Worah, D. and Sheth, A. (1997). Transactions in Transactional Workflows. In *Advanced Transaction Models and Architectures*, pages 3–41.