Liu, C., Zhou, X., & Orlowska, M. E. (1998). Issues in workflow and web-based workflow systems.

Paper presented at *'World Wide Web: technologies and applications', the 1998 Asia Pacific Web Conference (APWeb98), Beijing, China, 27–30 September 1998.*

Issues in Workflow and Web-based Workflow Systems

Chengfei Liu
School of Computing Sciences
University of Technology, Sydney
NSW 2009, Australia
liu@socs.uts.edu.au

Xiaofang Zhou
Mathematical and Information Sciences
CSIRO, GPO Box 664, Canberra
ACT 2601, Australia
Xiaofang.Zhou@cmis.csiro.au

Maria E Orlowska
Department of Computer Science and Electrical Engineering
University of Queensland
QLD 4072, Australia
maria@it.uq.edu.au

Abstract

Workflow management systems (WfMSs) have been used for organizational business process re-engineering and automation in various application domains. The use of ubiquitous web as infrastructure of WfMSs widens the applicability of workflow technology to almost every organization. Workflow technology is becoming the core technology for enterprise computing. However, many limitations remain in existing workflow management systems for supporting more demanding applications. In this paper, we address several issues in current workflows management: workflow modeling and verification, transactional support of workflows, and workflow evolution. We also discuss the use of web technology as an infrastructure for supporting WfMSs.

1 Introduction

The requirement for streamlining business processes through re-engineering and automation has influenced research and development in workflow systems. Workflow management systems (WfMSs) [25, 13, 27] have been used for re-engineering and automating business processes to various degrees in application domains such as telecommunications, finance and accounting, manufacturing, office automation, and healthcare. The success of WfMSs has been driven by the need for businesses to stay technologically advanced in the ever-increasing competition of global markets.

A workflow is used to model and automate a business process by coordinating a set of tasks that are connected in order to achieve a common business goal. Typically, the tasks in a workflow systems are executed by different processing entities in a heterogeneous, distributed and autonomous environment. Each task defines a logical step that contributes towards the completion of a workflow, it may be completed by human, by an application system or by both of them. A workflow management system (WfMS) provides a set of tools for workflow model specification, workflow enactment, administration and monitoring of workflow instances.

Due to the diversity of business processes, the workflow market has been divided into four main segments - *administrative, ad-hoc, production* and *collaborative* - based on the value of the process being managed and whether the process being managed is repetitive [15]. High value business processes are those that represent a significant cost saving or revenue-gaining opportunity and are usually tightly coupled with other business critical applications. In contrast the low value processes are those inherent in everyday work. The other axis measures how repetitive the process being

managed is - do many instances of work follow the same rules or are they relatively unique.

Figure 1 shows the four segments of existing workflow products though the vendors refuse to be pigeonholed into any one segment. Typical applications of each segment are also shown in the figure.

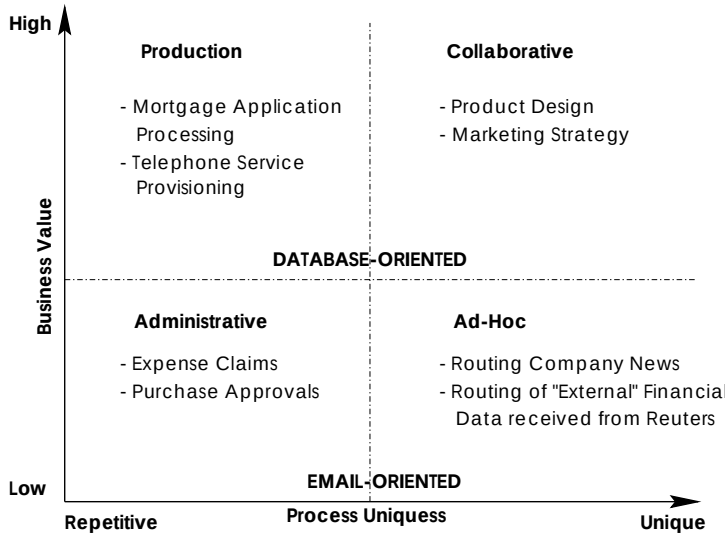

Figure 1: Segmentation of Workflow Market

With the advance of internet and intranet technology, the use of Web as infrastructure for supporting WfMSs widens applicability of workflow technology to most organizations. For an organization which wants to adopt WfMS technology, it is a significant investment to interconnect computers and to overcome the problems caused by the heterogeneity of computer systems and information systems. This problem is particularly frustrating when different organzations are involved in a workflow process, or the tasks assigned to those people who are not attached to specific sites. Advances in the internet technologies seem to be promising in providing solutions to these problems. First, nowadays most computers, being major computer servers in the corporate headquarters or personal computers at staff's desktops or homes, are readily connected to the internet or intranet. Information can be exchanged among different types of computers which are located in different continents, or just across the corridor. Second, the popularity of the Web technology means that anyone, with proper permission, can access a software system using any type of computer from anywhere. Different types of gateways systems and "wrappers" can connect a Web server to information sources from file systems, database management systems to legacy information systems. Another advantage from the ubiquity of the internet is that customers, an important part of business processes but cannot be integrated into WfMS before, can finally be part of the integrated process. Third, recent developments in internet security make possible to conduct electrical commerce on the internet. Thus, the level of protection required to support WfMS on the internet is already available from the off-the-shelf commercial products. Finally, convergence of the two major modern distributed technologies, the Internet and CORBA (Common Object Request Broker Architecture), makes much desired services such as trading and transaction processing support available on the internet. This will certainly greatly simplify the work to support WfMS on the internet.

In spite of the proliferation of workflow products, workflow technology is still immature to be able to support more demanding applications. Current workflow products are most appropriate for office environment with emphasis on coordinating human activities, facilitating document routing, imaging and reporting. Some of the apparent weaknesses of current workflow technology include: the lack of a clear theoretical basis, no commonly acceptable models, undefined correctness criteria, limited support of transactional behaviors, limited support of workflow evolution. In this paper, we address some of the above issues. We also discuss web-based workflow system implementation.

- Workflow modeling and verification
  To provide a good and correct design, to avoid design errors at early stage, a modeler should be able to capture easily semantics that might be inherent in the applications, and to verify easily the correctness of specification. Most workflow products lack a verification facility at this stage.
- Transactional workflows
  As the workflow concept has evolved from the notion of process in a business community, not in the database community, transaction-like behaviors of workflows have not been well supported by current workflow products. In fact, incorporation of transactional aspects into workflow applications is very useful for production workflow applications. Workflows with transaction-like behaviors are referred to as transactional workflows [29, 20].
- Workflow evolution
  Business processes are dynamic in nature, therefore, the workflow models which represent the business processes have to evolve. In addition, workflow instances tend to be long lasting, when a change happens to the model, *handover* of these instances is worthy studying. As far as we know, workflow technology to date supports very little for workflow evolution.
- Web-based workflow systems implementation
  Web technology has been deployed for workflow implementation [8, 5]. Some WfMS vendors such as Action Technologies, FileNet have already provided web-enabled products. However, due to the primitive nature of the web, many issues remain in the web-based workflow systems implementation, such as security, transactional support.

The rest of the paper is organized as follows. In Sections 2 - 4, we address workflow modeling and verification, transactional aspects of workflow systems, workflow evolution, respectively. Web-based implementation of workflow systems is addressed in Section 5. Section 6 concludes the paper.

2 Workflow Modeling and Verification

Workflow management is regarded as *programming in the large* for enterprise computing. Therefore, a precise workflow specification language and effective verification mechanism are important to prevent errors of workflow applications at design stage. A workflow models different aspects of a business process. These include: *control flows, data flows, temporal constraints, transactional requirements*, etc. Almost all modelers of workflow products support explicitly some sort of control flows. [22] gives a comparison of several workflow modelers with the control flow constructs provided by Workflow Management Coalition (WfMC) [25]. The constructs include: sequential, AND-join, AND-split, OR-join, OR-split, iteration, sub-process. Data flows and temporal constraints are usually specified implicitly. Some workflow products such as IBM's FlowMark [9] support explicit specification of data flows. Temporal constraints are supported by Action Workflow [2].

The objectives of a workflow modeling language are two-folds: expresiveness and easiness for verification. The workflow modelers of most workflow products are sufficiently expressible and also flexible for specifying workflow applications. However, lots of modelers use implicit constructs which make the verification very difficult. In that case, it is possible to easily get into error situations while building complex workflow models. Therefore, the study of conceptual workflow modeling and verification is useful for understanding the business processes and their accurate mappings onto workflow models.

Verification problems of control flows in workflow modeling has been studied in [32, 30]. A graphical workflow specification language is proposed for workflow conceptual modelling [30]. The language includes four types of modeling objects: task, condition, synchronizer, and flow. The flows are used to link the first three types of objects to build the specification of a workflow model.

- Task – A task is a logical step or description of a piece of work that contributes towards the accomplishment of a workflow model. It can represent both automated activities and human activities. Tasks are performed by assigned processing entities. A workflow model is basically used to specify the coordination requirements among tasks. The actual semantics of tasks is beyond the scope of workflow models.
- Condition – A condition is used to represent alternative paths in a workflow model depending on a conditional value.
- Synchronizer – At certain points in workflows, it is essential to wait for the completion of more than one execution path to proceed further. A synchronizer is used for this purpose.
- Flow – A flow defines the connection between any two objects, other than flows, in the workflow specification. It shows the flow of control or data from one object to another.

A workflow model can be represented as a *workflow graph* using these modelling objects, where nodes of the graph can be tasks, conditions and synchronizers and links of the graph are flows. Restriction is placed in constructing a correct workflow graph. Only a limited yet relatively complete set of constructs are supported by the language. They are *Sequential, Exclusive OR-Split (Alternative), Exclusive OR-Join, AND-Split (Parallel), AND-Join (Synchronization), Sub-process, Iteration, Start/Stop*. The language use a condition construct to express the OR-split semantics and the construct always results in two exclusive yet complete branches. The OR-join is expressed as multiple incoming flows and one and only one flow can be activated thus also has the exclusive semantics. These two constructs have clear and unique semantics.

As compared in [22], this language is more strict than the modeling language provided by some workflow products, say FlowMark. In FlowMark, the OR-split is expressed as multiple outgoing control connectors attached with transition conditions. The OR-join is expressed as multiple incoming control connectors with start condition valued ANY. Both OR-split and OR-join in FlowMark are inclusive, i.e., allowing more than one outgoing and incoming control connectors. The inclusive OR-split construct in FlowMark also allows irrelevant transition conditions specified on outgoing control connectors, therefore, its semantics can be many, from none of the conditions are satisfied (which results in part of a process or whole process instance terminate) to all of them are satisfied. In fact, the uncertain semantics include three orthogonal aspects:

- both exclusive and inclusive (overlaping or implied)
- both complete and incomplete partition
- both relevant and irrelevant

As such, the flow of control is buried in conditions. Same syntactical control structure can result in very different control flow. This brings troubles for reasoning the correctness of a workflow model. This brings inaccuracy of workflow model as well because same construct is used for different semantics.

In addition, allowing stop at every activity in FlowMark is also problematic, though sometimes it is flexible. In fact, an unexpected stop can be an error in user's requirement. Unfortunately, such a requirement can not be specified as an error in FlowMark. In FlowMark, stop at any activity is always right. Obviously some semantics are missing here. This again brings trouble for early detection of errors.

In [30], a set of correctness constraints of workflow graphs have also been identified and verification algorithms have been proposed for verifying the syntactical correctness of workflow graphs specified using this language. For instance, all or none of the flows proceeding a synchronizer activate for all possible instances of a workflow. Only one or none of the flows leading to a task or a condition activates for all possible instances of a workflow. The first rule eliminates the possibility of a *synchronizer deadlock*. The second rule eliminates the possibility of an *unintentional multiple execution*.

The verifications of other coordination aspects (i.e., data flows, temporal constraints, etc.) need to be investigated. The results of verification study are useful for frontend tools of workflow modelling.

3 Transactional Workflows

Two approaches are used in a multi-system application: process-centric and data-centric. A WfMS takes the process-centric approach as the workflow concept has evolved from the notion of process in a business community. The process aspect (business processing logic) of multi-system applications is highlighted in workflow systems. A workflow model uses a declarative specification language to facilitate a seperation between the application code and task coordination structure, i.e., control flow and data flow between different tasks of the application. However, the transactional aspect (data processing logic) of multi-system applications have not been well supported by current workflow technology. This restricts the wide applicability of workflow management systems to enterprise computing, especially for the high value product workflow applications.

Current transaction-oriented systems, on the other hand, take data-centric approach, the transactional aspect (ACID properties) is well-supported in these systems. The well known two-phase commit (2PC) protocol is supported by TP-monitors and multidatabase systems. However, transaction-oriented systems provide only limited support for managing the control and data flows of multi-system applications. For instance, TP-monitors try to provide mechanisms for task coordination in the form of *queued transactions*, however, the task coordination structure is still scattered through the application code.

The applicability of transaction-oriented systems to multi-system applications is restricted as most such systems put strict requirements on constituent systems. For example, 2PC protocol requires the *prepared* state of all subtransactions (constituent systems) but one visible to the coordinator. In fact, many constituent systems used in multi-system applications are likely lecacy systems or unable to provide 2PC services. In addition, the long-duration of some multi-system applications will make 2PC non-applicable.

As application systems used for executing tasks in workflow applications were designed for stand-alone operation, they normally do not provide the information and services that would be necessary to execute the distributed transactions, therefore, the incorporation of the transactional aspect into workflow need to be addressed specifically. Workflows with the transactional aspect are referred to as transactional workflows. Transactional workflows address application specific and user-defined correctness, reliability, and functionality requirements. They share the objectives of some advanced transaction models [7] about the selective relaxation of transactional properties of business processes based on application semantics.

One natural approach towards transactional workflows for database researchers is to enhance advanced transactional models [7] by incorporating workflow concepts. For example, ConTracts [28] were proposed as a mechanism for grouping transactions into a multitransaction activity. A ConTract consists of a set of steps with ACID properties and an explicitly specified execution plan called script. An execution of a ConTract must be forward-recoverable. Relaxed atomicity is also provided in ConTracts. However, most advanced transaction models are often too rigid for workflow applications, they often provide a predefined set of transactional properties which may not be flexible for workflow applications.

Another approach is to utilize a workflow model as the basis, and complements it with transactional features such as reliability, consistency, and other transaction semantics. We think this is the appropriate approach for transactional workflows. Efforts have been put to incorporating transaction aspect into workflows [4, 29, 19, 21]. In [4], Breitbart et al. proposed to merge application-centric and data-centric approaches for transaction-oriented multi-system workflows. They studied various dependencies between tasks as well as semantic properties of the workflow tasks and tried to combine workflow scheduler and semantic workflow scheduler together. [29] presented a transactional workflow model by extending their work on multidatabase transactions.

In [19], we propose a transactional model which incorporate relaxed ACID transaction properties into workflow model. The model allows both transactional tasks and non-transactional tasks such as legacy and non-DBMS applications be included in a workflow. The properties such as commit protocal, compensatability of tasks involved in a workflow, together with the relaxed requirements of failure atomicity of the workflow are studied and used to determine the correctness and schedule of the workflow. The model is designed to meet the application-specific transactional requirements.

So far, the work done in transactional workflows is still far from practical use. However, one result out of transactional workflows research is promising, i.e., backward recovery support of workflows. Reliability is important to workflow systems as failures could occur at various stages within the life-time of a workflow instance. When an erroneous situation occurs, the effects of part or whole failed instance need to be eliminated. As a workflow tends to be long lasting, the concept of compensation [14, 11] is borrowed from advanced transaction models for backward recovery of workflows [19, 18, 21, 23]. By compensation, the effect of an executed task T is eliminated by executing a compensating task CT of T which can *semantically undo* the task T after execution of T (i.e., T has been committed in terms of a transaction). For example, the compensation of a reservation is a cancellation, and the compensation of a deposit is a withdrawal. Based on compensation, we have implemented a backward recovery mechanism on the top of IBM's FlowMark [18].

However, designing compensating tasks for tasks is a non-trivial job, requiring knowledge of the semantics of the workflow applications. In fact, not every task is compensatable in the sense that the *forcibility* of the reverse operations of the task are not always guaranteed by the application semantics. In addition, the isolation requirement of resources may make the task difficult to compensate. In [23], we carefully investigate the properties of shared resources and tasks which may be performed on these resources. A new concept called *confirmation* is introduced to overcome some of the non-compensatability problem of tasks. A framework for incorporating compensation and confirmation has been proposed.

4 Workflow Evolution

In a fast-changing environment, an organization may constantly refine its workflow models to remain competitive in the market, to meet customers' new requirements, to change business strategies, to improve performance and quality of services, to benefit from changes in technology, etc. Therefore, workflow evolution is inevitable. Examples of such an evolution can be a policy change of a mortgage application processing, or a replacement of some old parts with new parts of an automatic assembly in manufacturing setting. Two aspects are related to the evolution of a workflow model. First, the old specification of a workflow model needs to be changed to a new specification correctly. This is the static aspect of the evolution. Second, as a business process tends to be long lasting, whenever a workflow model changes its specification, there may exist a set of running instances of the old specification. Instances executing at different stages with different execution history may be changed over differently. Obviously, it is not desired to handover these running instances case by case as the number of runing instances can be large at the time of evolution. Therefore, how to handover running instances of an evolving workflow is an interesting and challenging issue.

Currently, workflow evolution has not been sufficiently supported by workflow products. Only very primitive policies are supported by some workflow products such as *Forte Conductor* [10]. Other workflow products such as *InConcert* [16] support dynamic workflow adaptation at the instance level [31]. Schema evolution has been widely addressed in the field of Object-Oriented Databases and Software Processes [3, 17]. However, little work has been done in addressing the problem of workflow evolution, particularly the dynamic aspect. In their paper [6] Casati et al. have presented a workflow modification language that supports modification of a workflow model (schema). They have also discussed the case evolution policies and have devised three main policies to manage case evolution: *abort* – to abort all running instances and to start new created instances following new specifications, *flush* – to finish all running instances first and then to allow

new instances to start following new specifications, and *progressive* – to allow different instances to take different decisions. Though the progressive policy is further discussed in their paper, the fine granularity of the case evolution policy specifications has not been addressed. We have specifically addressed the dynamic aspect of workflow evolution [24]. As whether or not the running instances shall evolve according to the new specification and how they evolve depend on an evolution policy which is specific to the business process which the workflow model represents for, our focus was put on the specification and facilitation of evolution policies, or *handover* policies, as we called.

We view the evolution of a workflow model as a process which consists of three steps: (1). to modify a workflow model from its old specification to its new specification. (2). to specify a handover policy for handing over the running instances of the workflow model to be evolved. (3). to apply the handover policy. A workflow model modification can be done by applying a series of modification primitives using a workflow model modification language. The handover policy is formulated based on the old and new specifications of a workflow model. When specifying a handover policy, we assume that a specifier has knowledge of both old and new specifications of the workflow model as well as their difference. Step 1 and Step 2 are performed at build-time, while only Step 3 is performed at run-time.

In our study [24], we designed a handover specification language. The objectives of the language is effective yet simple. As when a handover policy is applied to an evolution of a workflow model (i.e., from its old specification to its new specification), the running instances may be executing at any task of the old specification. What is worse, different instances can take different paths to the same task. Therefore, different instances may require different handover strategies. No matter how complex the situation can be, the language should be expressive enough for specifying all a workflow administrator wants to specify. Obviously, if all the possibilities need to be specified explicitly, it can be cumbersome, even not applicable to large workflow models. Therefore, simplification of specification must be considered. Fortunately, in practice, a workflow administrator is only interested in some key points where turning actions need to be taken. Using some default and grouping specification, the specification of a handover policy can be greatly simplified.

A set of handover statements are used to represent a handover policy, with each statement reflecting a key point. In each statement, three handover aspects of a running instance are specified:

- current position – indicating current executing task of a running instance;
- history – indicating the traversed paths of a running instance by conditional value;
- action – indicating the action to be taken.

Two types of turning actions are especially important to workflow instance handover. One action is *rollback*. It is used to *semantically undo* some work so that the running instance can comply with the new workflow specification. The other is *change-over*. It is used to migrate the execution of a running instance (or a path of it) to follow the new specification. There is another action called *go-ahead* which may not be explicitly specified. If no handover statement is defined on a task, the default handover action after the execution of the task is going ahead.

Sometimes, a turning action at the current task is decided based on the history (i.e., the traversed paths) of a running instance. This is supported by the conditional turning by representing the history information in a conditional value. Besides the history information, other semantic information (e.g., time) can also be specified in the condition to facilitate flexible handover.

The correctness and verification issues of handover policy specification are also discussed in the paper [24]. A framework which can be used for realizing handover of running workflow instances has also been put forward based on current workflow technology. Currently, we are prototyping the framework. We are also furthering study on multiple version support of workflow evolution.

5 Workflow and the Internet

The internet provides ideal support to WfMSs in two major aspects. One is to provide distributed systems infrastructure for workflow execution and administration. The other is to provide a web interface such that the user of a WfMS can define a workflow using a standard web browser (e.g., Netscape Navigator and Microsoft Internet Explorer) to specify a workflow, to use the web browser to perform certain tasks in a workflow, as well as to use the web browser to monitor and administrate workflow execution.

The trend of combining Internet and WfMSs is evidenced by many pioneer WfMS systems which make use of the web in one way or another [26, 5]. The first wave of adopting the web in WfMS resulted in a number of web-enabled systems, which provide a web interface to conventional WfMSs, in a way similar to providing a web interface to a database system or a legacy information system. The current tread is, however, to build a WfMS completely on top of the internet, using not only the web as its interface, but also relies solely on the internet for communication support. A survey about the application of the web in WfMSs can be found in [26].

Now we discuss in greater detail how the internet technology can be used in implementing WfMSs. The human-interaction components of a WfMS, such as for the user to define a task, to give input to tasks or to perform certain monitoring and administrative work, can be done within a web browser, using either HTML forms, or Java Applets when some processing (e.g., input data verification) need to be done of the client side. A web server is, obviously, a central part of a web-based WfMS, which can be used as the repository for workflow management tools such as workflow definition. It is also a place to hold worklists for multiple users to share. CGI (Common Gateway Interface) is used for a web server to invoke services, which can be an automated task performed by a program (including a legacy information system), or queries to a database management system. Database access can be implemented using CGI (with databases APIs such as SQL and ODBS), or the JDBC interface.

In an Internet Marketplaces project [1] currently under development at CSIRO Mathematical and Information Sciences, we have used both HTTP and CORBA as communication support for execute distributed tasks on the internet. We find that HTTP is low-cost, simple but sufficient for such applications. On the other hand, CORBA has several advantages. First, it supports stateful connections, in contrast to HTTP which only supports stateless connections (however, we find for workflow execution only coarse-grain interactions are necessary between tasks, thus stateless connection is quite acceptable). Second, CORBA provides much powerful support at distributed programming level (via, for example, IDL), while HTTP is simply a communication protocol. One benefit of CORBA here is that complex objects can be exchanged between tasks, saving the potential costly procedure to flatten them and pass them as plain text as in HTTP. The third, CORBA promises many other services, such as Object Transaction Processing, which can be useful for supporting transactional workflows.

Security is a major concern for web-based WfMSs. This is one reason that most organisations are only willing to use Intranet. However, recent advances in web security, in terms of both securing the web servers and the users' computers and securing information in transit [12], greatly reduce the rick of conducting business on the internet. For many WfMSs which based on membership paradigm (i.e., all users are pre-registered users), protocols such as PGP can provide sufficient protection.

6 Conclusion

Workflow technology is the most appropriate technology for enterprise computing. The web-enabled workflow systems makes it more applicable for most organizations. Many organizatons are shifting their data-centric approach in the context of the information systems technology and solutions to a process-centric workflow approach. However, the workflow technology is immature in certain aspects. In this paper, several limitations in current workflow systems have been discussed,

these include workflow modeling and verification, transactional workflows, and workflow evolutions. Issues in current web-based workflow systems have been addressed as well. Due to the ubiquitous nature of the web, it is especially important to study the implementation issues of web as a infrastruture for supporting workflow management systems.

References

[1] D. Abel, V. Gaede, K. Taylor, and X. Zhou. Smart: Towards spatial internet marketplaces. *GeoInformatica*. to appear.

[2] TOTAL FrameWork ActionWorkflow. *Process Builder User Guide*. Action Technologies, Inc., http://www.cincom.com, 1996.

[3] J. Banerjee, W. Kim, H-J. Kim, and H. Korth. Semantics and implementation of schema evolution in object-oriented databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 311–322, 1987.

[4] Y. Breibart, A. Deacon, H.-J. Schek, A. Sheth, and G. Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. *SIGMOD Records*, 22(3), 1993.

[5] T. Cai, P. Gloor, and S. Nog. Dartflow: A workflow management system on the web using transportable agents. Technical report, Dartmouth College, 1997. URL:http://www.cs.dartmouth.edu/reports/Cai,Ting.html.

[6] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. In *Proceedings of the 15th ER Int. Conf.*, pages 438–455. Lecture Notes in Computer Science, Vol. 1157, Springer, 1996.

[7] A. Elmagarmid, editor. *Database Transaction Models For Advanced Applications*. Morgan Kaufmann, 1992.

[8] A. Sheth et al. Supporting state-wide immunization tracking using multi-paradigm workflow technology. In *Proceedings of the 22nd VLDB Conference*, Mumbai, India, 1996.

[9] IBM FlowMark. *Modeling Workflow*. IBM, February 1996. Version 2 Release 2.

[10] Forte. *Forte Conductor Process Development Guide*. Forte Software, Inc., 1994.

[11] H. Garcia-Molina and K. Salem. Sagas. In *Proceedings of the ACM Conference on Management of Data*, pages 249–259, 1987.

[12] S. Garfinkel and G. Spafford. *Web Security and Commerce*. O'Reilly, 1997.

[13] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3:119–153, 1995.

[14] J. Gray. The transaction concept: Virtues and limitations. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–154, Cannes, France, 1981.

[15] Butler Group. *Workflow: Integrating the Enterprise*, June 1996.

[16] InConcert. *InConcert Technical Product Overview*. InConcert Inc., January 1997.

[17] M. Jaccheri and R. Conradi. Techniques for process model evolution in EPOS. *IEEE Transactions on Software Engineering*, 19(12):1145–1156, December 1993.

[18] B. Kiepuszewski, R. Muhlberger, and M. Orlowska. Flowback: Providing backward recovery for workflow management systems, June 1998. Demonstrated at SIGMOD'98.

[19] D. Kuo, M. Lawley, C. Liu, and M. Orlowska. A general model for nested transactional workflows. In *Proceedings of the International Workshop on Advanced Transaction Models and Architectures*, pages 18–35, 1996.

[20] D. Kuo, M. Lawley, C. Liu, and M. Orlowska. A model for transactional workflows. In R. Topor, editor, *Seventh Australasian Database Conference Proceedings*, volume 18, pages 139–146, Melbourne, Australia, 1996. Australian Computer Science Communications.

[21] F. Leymann. Supporting business transactions via partial backward recovery in workflow management systems. In *Proceedings of BTW'95*, pages 51–70, 1995.

[22] C. Liu. Comparison of workflow modelers. Technical report, Distributed Systems Technology Centre, University of Queensland, QLD, 4072, June, 1997.

[23] C. Liu and M. Orlowska. Confirmation: Increasing resource availability for transactional workflows. Technical report, Distributed Systems Technology Centre, 1997.

[24] C. Liu, M. Orlowska, and H. Li. Automating handover in dynamic workflow environments. In *Proceedings of Advanced Information Systems Engineering 10th International Conference (CAiSE*98)*, Pisa, Italy, June 1998.

[25] Workflow Management Coalition Members. *Glossary – A Workflow Management Coalition Specification*. Workflow Management Coalition, November 1994. Softcopy available via: http://www.aiai.ed.ac.uk/project/wfmc/.

[26] J. Miller, D. Palaniswami, A. Sheth, K. Kochut, and H. Singh. Webwork: Meteors web-based workflow management system. *Journal of Intelligent Information Systems*, 10(2), 1988.

[27] C. Mohan. Tutorial: State of the art in workflow management system resea rch and products, March 1997. Presented at DASFAA'97.

[28] A. Reuter. A means for extending control beyond transaction boundaries. In *Proceedings of the 2nd International Workshop on High Performance Transaction Systems*, 1989.

[29] M. Rusinkiewicz and A. Sheth. Specification and execution of transactional workflows. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*. Addison-Wesley, 1995.

[30] W. Sadiq and M. E. Orlowska. On correctness issues in conceptual modeling of workflows. In *Proceedings of the 5th European Conference on Information System*, 1997.

[31] A. Sheth and K. Kochut. Workflow applications to research agenda: Scalable and dynamic workflow coordination and collabration systems. In *Proceedings of the NATO ASI on Workflow Management Systems and Interoperability*, August 1997.

[32] A.H.M. ter Hofstede, Maria E. Orlowska, and J. Rajapakse. Verification problems in conceptual workflow specification. In *Proceedings of the 15th ER Int. Conf.*, pages 73–88. Lecture Notes in Computer Science, Vol. 1157, Springer, 1996.