An Overview of the Exotica Research Project on Workflow Management Systems*

C. Mohan G. Alonso R. Günthör M. Kamath B. Reinwald

IBM Almaden Research Center, San Jose, CA 95120, USA
http://www.almaden.ibm.com/cs/reports/workflow/exotica_home_page.html
mohan@almaden.ibm.com, alonso@inf.ethz.ch
Roger.Guenthoer@informatik.uni-stuttgart.de, kamath@freya.cs.umass.edu
reinwald@almaden.ibm.com

1 Introduction

Workflow is probably one of the most exciting areas of research that has emerged in the past few years. Workflow concepts and ideas have been around in one form or another for a long time: *computer supported cooperative work*, *forms processing*, *cooperative systems*, *office automation*, etc. However, only recently the technology and know-how required to implement commercial systems have become available. In general, workflow management systems (WFMSs) are used to coordinate and streamline *business processes*. These business processes are represented as *workflows*, i.e., computerized models of the business processes [13], which specify the individual *activity* steps, the order and the conditions in which the activities must be executed, the flow of data between activities, the users responsible for the execution of the activities, the tools to use with each activity, etc. A WFMS is the set of tools that allow the design and definition of workflows, their instantiation and controlled execution, and the coordination and integration of heterogeneous applications within the same workflow [13]. Users interact with a WFMS by accessing their individual *worklists*, where they can find the activities for which they are responsible without necessarily being aware of the higher level processes to which the activities belong. A crucial point to understand workflow management systems is their dependency on a variety of technologies, from databases to distributed processing. When designing a WFMS, the challenge is to build a feasible and common working environment in which all these technologies are integrated in a flexible and easy to use fashion. This integration aspect is precisely what has made WFMSs so elusive to date and explains the limited success of earlier attempts to provide support for cooperative work. With the first generation of commercial WFMSs [12], there are still many integration issues that remain to be solved, but there is no doubt that in the future WFMSs will be pervasive in large corporations.

In this paper, we present the Exotica research project currently in progress at the IBM Almaden Research Center. One of the goals of the project is to bring together industrial trends and research issues in the workflow area. It is for this reason that we have focused on a particular commercial product, *FlowMark*, IBM's workflow product [15, 16, 19, 20]. However, our results are easily generalized to other WFMSs since FlowMark's model is similar to that proposed by the Workflow Management Coalition [13]. In particular, the rest of this paper contains a high-level overview of our research in six specific areas that are not product specific. The list of these areas is not, by any means, exhaustive. There are still many issues that remain open. We also discuss the relationship between WFMSs and transaction processing monitors.

*This work is partially supported by funds from IBM Hursley (Networking Software Division), and IBM Boeblingen and Vienna (Software Solutions Division). Even though we refer to specific IBM products in this paper, no conclusions should be drawn about future IBM product plans based on this paper's contents. The opinions expressed here are our own.

2 Large Scale Workflow Systems

Before discussing workflow management, it is necessary to put in perspective the applications addressed by such systems. Some of the common goals of a WFMS are to achieve better performance of business processes, better quality, enhanced effectiveness, enterprise wide coordination and monitoring, etc. As we believe it is the case with most WFMSs, FlowMark addresses the coordination of *large* scale business processes. With respect to this, there are certain similarities with DBMSs. Small DBMSs for personal computers have an undeniable value. However, interesting research and industrial strength products relate to issues such as query optimization, performance, data caching, data mining, schema evolution, or triggers, which only become interesting in large DBMS scenarios. For small applications, it is possible to obtain similar or even more benefits by using a less sophisticated cooperative tool rather than a full-fledged WFMS. However, when the size of the application grows, and the number of users, sites, and processes increases, only a WFMS is suitable for the task. It remains to be determined how "large" is a large application. To give an idea of the magnitude of the problem, consider some of the studies that have been done with FlowMark. In one instance, the number of business processes to be executed concurrently during the course of a month is 300,000. In another case, the problem is to link together more than 4,000 sites geographically distributed over an entire country and coordinate the work taking place over heterogeneous platforms at those sites. A third example involves more than 100,000 users working concurrently on the completion of processes. With these figures, issues that were not previously considered as related to workflow become key aspects to the success of a commercial system. Failure handling, continuous availability, navigational flexibility, and replication, to name a few, are no longer nice features but crucial components of the overall system. It is within this framework that our research is inscribed.

3 Research Issues in Workflow Management

The Exotica project started out focussing on six major research areas. Each of them reflects a distinctive need for WFMSs but, in most cases, they are related to each other and successful solutions will have to integrate aspects from all of them. The list of these areas is by no means exhaustive, there are many other issues that remain open.

3.1 Failure Resilience in Distributed WFMSs

Given its goals, the architecture of a WFMS is necessarily distributed in the sense that its components will reside in different and heterogeneous machines. Furthermore, given its relevance in the control of the business processes of a corporation, the system must be continuously available. Hence, each component must be able to deal with local and communication failures, and the design must be robust enough to avoid stopping the execution of processes even in the event of failures. Such requirements define a wide range of failure handling capabilities that the system must support to be commercially viable. Part of our research has been to analyze the possible failure scenarios and design methods to handle them [1, 2, 18].

In the first place, the execution of an activity within a process involves several components: the database server where the workflow-process-related data is stored, the workflow server that determines where the activity is going to be executed, the client interacting with the user, and the client interacting with the actual application performing the activity. All of these components interchange information before, during and after the execution of the activity. This information should not be lost, since this may imply that the results of the execution are not recorded, or that the command to start executing the activity never reaches its destination. Hence, some form of *coordination protocol* must be used amongst all these components to guarantee that all agree on the status of a particular activity. In general, the DBMS in which the data is stored is transaction based. In FlowMark, for instance, the data is stored in ObjectStore, an object-oriented DBMS. ObjectStore does not provide either *hot standby* support or a *prepare-to-commit* interface, which makes it complicated to reach a point in the execution where data is guaranteed not to be lost. Moreover, the applications invoked by the WFMS may not be aware of the WFMS's existence, thereby becoming uncooperative and committing work regardless of the state of the WFMS. We are studying several approaches to these problems: using a persistent message mechanism, providing stable storage in each of the components, and designing a handshake protocol that ensures that information is almost never lost.

A second aspect of failure resilience is the possibility that a component fails while it is executing part of a process. If the failed component is the DBMS, replication can be used to minimize the impact of the failure, as discussed below. If the failed component is part of the WFMS, means must be devised to allow other components to reroute their operations to components that are still available. This implies using multiple connections among components. We have designed a new architecture for WFMSs in which the notion of clusters of servers is used to provide enhanced availability of the system [2, 18]. By using several clusters, each of them attached to a separate database, the impact of failures on overall performance and availability can be reduced. Since each cluster's database contains the same schema information such as process templates, role and staff assignments, and organization definitions, process instances can be run at any cluster. Thus, failures will affect only a subset of the process instances and will not prevent the users from starting new instances. The problem that still remains is how to handle the instances that were running in a failed cluster. This particular issue is addressed below when discussing replication in WFMSs.

3.2 Compensation and Navigation in Workflow Networks

A workflow network, or a process, is the representation of a business process. Its instantiation for execution is a process instance. Any process instance usually encompasses a wide range of activities and those activities involve several participants in the form of computer programs and individuals. At any given moment in time, there may be several instances of the same process being executed simultaneously, with possibly each one being at a different stage of progress. Since processes are defined in advance, it is not possible to evaluate all possible exceptions and error conditions beforehand. A WFMS must provide means to cope with such situations. Similar ideas are found in advanced transaction models [9] such as Sagas [10, 11] for backward recovery, i.e., how to undo the effects of certain operations, or in Flex transactions [7, 22, 31] for forward recovery, i.e., how to select different paths of execution. However, in WFMSs, there are certain errors that may force the user to abort the execution of a process instance, and therefore to lose work that has to be performed again when the process is restarted.

Forward recovery is guaranteed in FlowMark in the sense that if there are failures, the process is guaranteed to make progress. However, semantic failures are more difficult to address, unless the designer of the process was able to foresee them and introduce the appropriate checks in the flow of control. For backward recovery, FlowMark is being enhanced to provide a form of compensation that allows the user to specify *spheres of compensation* [21] to determine the scope and extent of compensation in case of failures.

Our current research is focused on how to provide a flexible mechanism to navigate through the flow of control of a process in either direction, i.e., forward progress or compensation, and allowing to switch directions several times as necessary during the execution of a process.

3.3 High Availability through Replication

High availability is a key requirement of a WFMS. Failures should be transparent to the users and should have minimal impact on the normal functioning of the organization. This can only be achieved through replication of the process instance information. As in DBMSs [25], we use a primary/backup architecture in which process instances run on a primary server, while all actions are also recorded at the backup so that if the primary were to fail, then the backup server can take over the execution of those same instances. As in DBMSs, replication has a high cost in terms of synchronization and lower throughput. Given the large number of processes involved in the application, replication of every single process instance may not be cost effective. Hence, we define three priority levels with respect to replication. A *hot stand-by* process is fully replicated, i.e., all changes to its data are performed both in the primary and in the backup databases. A *cold stand-by* process is also replicated, but the backup contains only the messages with the information regarding the different steps taken in the execution of the process. When the backup needs to take over the execution of the process, it must first replay those messages to bring the process state up-to-date. This will delay resuming the execution but it reduces the overhead of maintaining the replicated copies. Finally, a *normal* process is not replicated at all. The only guarantee FlowMark provides for these processes is forward recovery, i.e., when the server comes up again, execution will be resumed from where it was left off.

As in DBMSs, besides the replication scheme, it is also necessary to deal with additional problems such as dynamic configuration of the system, incorporation and exclusion of new servers without interrupting the system,

message duplication, and so forth. We have addressed all these issues and proposed a replication mechanism for WFMSs that greatly improves the availability of the system while avoiding excessive overhead [18].

3.4 Mobile Computing

WFMSs must coordinate users distributed over a wide geographic area. The basic idea being that the processes to be executed are defined and controlled in a centralized server, while the users can execute parts of these processes at remote clients. Each step of a process can be executed anywhere in the system, but after its completion, the results are communicated to the server which in turns prepares the next steps for execution. This is the most common approach in existing systems, and simplifies many problems such as synchronization, concurrency and monitoring, but forces the users to remain *connected to the server* while performing multiple tasks. An increasingly large set of users may not be satisfied with this mode of operation, given the widespread use of portable computers and home computers for office work.

For these users, the common way of operation is to load data in their laptops or desktops by briefly connecting with a server in the office. Then they disconnect from the server, work on that data and, after a few hours or few days, reconnect to the server to transfer the results of their work. This has been identified as one of the most common modes of computing that will occur in the future [17]. Note that this includes not only mobile terminals but also desktops or any other computer type connected to the server only occasionally via a modem. Mobile or nomadic computing greatly expands the scope of an organization's distributed computing infrastructure. However, from the workflow management point of view, it becomes more difficult to coordinate the work of many users. Also, note that while WFMSs are tools for cooperation, portable computers are generally viewed as tools for individual work.

To address this problem, we have proposed a design, called Exotica/FMDC, for disconnected client operation based on FlowMark [3]. Among other constraints, the design had to provide support for disconnected clients with minimal changes to the semantics of a business process and its implementation, effectively allowing users distributed over a wide geographic area and working with heterogeneous resources to cooperate, while preserving their mobility and independence. In particular, we define the semantics of loading work to a mobile, disconnectable computer by introducing the notion of *locked activities* and the *user's commitment* to eventually execute them. Locked activities stay within the user's machine unless they are unlocked. The feasibility of our approach was initially proven by outlining the implementation issues for FlowMark. We have more recently completed prototyping Exotica/FMDC.

3.5 Distributed Coordination

To further enhance the availability and failure resiliency of a WFMS, it is possible to design it entirely as a distributed system. This implies that processes are transferred from site to site as their execution progresses, without a centralized server keeping all the relevant information. There are two possible approaches. One is to make a large package that contains all the information pertaining to the process and its execution and circulate this package across the different sites. This approach has several problems, mainly the size of the message and the need to maintain multiple copies of it if the system were to allow concurrent activities to take place. The other solution is to precompile the process definition to determine at which sites the different activities are to be executed. Once this has been done, the only information that gets transferred from site to site are the results of previous computations and the process state. Using this approach, failures can be made transparent by rerouting a process to different nodes. It also eliminates the potential bottleneck created by a centralized database and server. Exotica/FMQM [1] proposes a similar architecture in which persistent storage is replaced by persistent messaging. For persistent messaging, IBM has defined an application programming interface (API) standard called Message Queue Interface (MQI) [14], and a family of products called MQSeries that supports MQI [23]. MQSeries products operate on IBM and non-IBM platforms and they support the architected MQI. Communication takes place through named queues that do not require all participating programs to be available, i.e., up and running, simultaneously. Moreover, MQI is not sensitive to network transport protocol differences. The system we propose, *Exotica/FMQM*, FlowMark on Message Queue Manager, is a distributed WFMS in which a set of autonomous nodes cooperate to complete the execution of a process. Each site functions independently of the rest of the system, the only interactions between nodes being conducted through persistent messages which inform about activity completions. This approach, while enhancing the availability and resilience of the system,

has the problem of needing more complex mechanisms to monitor and audit the overall execution since there is no centralized server where this information resides. It also requires enhancements to the MQI to be able to assign an activity to more than one user but let only one of the users actually execute it.

3.6 Advanced Transaction Models

The goal of most advanced transaction models is to eliminate the constraints imposed by conventional DBMSs on new applications. The requirements of such new applications are completely different from the traditional data processing applications targeted by standard DBMSs. Hence, conventional DBMSs are unsuitable in these new environments. Several advanced transaction models have been proposed [9] but, to this date, most remain as theoretical constructs which have not been implemented. A reason for this state of affairs could be that advanced transaction models are ahead of the available technology and their time is yet to come. We believe, however, that the reason why these models are not being implemented has to do more with their inadequacy to operate in real working environments than with the available technology or application demands. Advanced transaction models are too database-centric, which provides a nice theoretical framework to work with but limits the possibilities and flexibility of the models. Since they tend to remain theoretical models with no implementation, there are a number of important design issues that are generally not discussed in the literature [24].

Paradoxically, WFMSs are tools to support distributed, heterogeneous environments very similar to those targeted by transaction models. However, the models being used are *workflow* models instead of transaction models. Compare the more than 70 vendors who claim to have WFMSs [12] with the almost absolute lack of commercial products supporting advanced transaction models. Workflow systems bear a strong resemblance to advanced transaction models both in their goals and modeling approach, yet they are quite different in that they address a much richer set of requirements than existing transaction models. As part of our research, we have analyzed the characteristics of workflow models and the notion of business processes by comparing them with existing transaction models [2, 4]. It is possible to show that the semantics of workflow models are, in general, richer than those of advanced transaction models and more apt to be used in commercial products. Hence, workflow models can be used to implement advanced transaction models. For instance, we have used FlowMark to implement Sagas [10] and Flexible Transactions [31]. Our basic goal was to provide synergy between advanced transaction models and workflow models, an interaction from which both sides may benefit. In doing so, we have developed a better understanding of the inherent limitations of the so-called "advanced models" and identified many points for improvement of WFMSs. Our approach is not another attempt to merge different transaction models into one or to provide a general framework to program advanced transactions [5]. Transactions are used as the accepted currency in the database community, but our goal is to show the expressibility and power of workflow models compared with what is currently available in the research literature [8].

From a purely advanced transactions point of view, our proposal differs from previous ones in that we use an existing commercial product that runs across different platforms to implement different transaction models. This allows us to draw conclusions from direct experience implementing such models in real, working environments. We see workflow models as the the next step after advanced transaction models. They complement many aspects of the latter and address an entirely new range of issues (role and staff assignment and resolution, worklist management, interaction with manual activities, etc.) that make them more suitable for building applications. As has been widely recognized, there is a lack of tools to scale from individual transactions to complex applications [30]. One reason being that advanced transaction models will never reach their technological maturity until they are interpreted in a much broader context. As we show in our research, business processes constitute such a context and workflow models are the tools required to support complex applications.

4 TP Monitors and Workflow Systems

A WFMS is a tool to control the execution of business processes based on representations of the organizational, informational, and functional aspects of an enterprise. In its origins, closely tied to *office automation*, workflow management was geared towards the system-enforced organization, processing, and automation of well structured cooperative work. However, automation efforts were, at the time, limited by the lack of appropriate technology. Nowadays, advances in the state of the art, such as new desktop operating systems or more sophisticated DBMSs and networks, have allowed the incorporation of many new technologies into the workplace. The result has been the partition of the enterprise into *islands of automation* that do not necessarily communicate with each other.

In this situation, WFMSs are seen as the key element to bring together all these technologies and provide an integrated cooperative environment in large organizations. This would explain the interest aroused and successes enjoyed by the first generation of commercial WFMSs.

Transaction processing monitors, TPMs, on the other hand, are a well established technology that has been around for almost 20 years [6]. The first generation of TPMs, e.g., IMS/DC and CICS, were single monolithic systems used in proprietary mainframe environments to achieve high transaction rates in applications such as airline reservations or finance handling. The second generation of monitors, e.g., ACMS, Encina, and CICS/6000, are more open and able to talk to heterogeneous resource managers under certain restrictions (such as support for standard *commit protocols*, for instance). In general terms, TPMs are very mature with regard to concurrency control and recovery and have solid foundations in the notion of *ACID transactions* as well as in high-end load balancing and failure handling.

In spite of addressing similar problems, the characteristics of WFMSs and those of TPMs are almost complementary. Workflow management lacks a clear transactional concept, and has problems with scalability, reliability, and failure handling. It provides, however, modelling of inter-application relationships (i.e., business process modelling), and persistent forward execution. WFMSs permit the invocation of applications operating in different execution environments with parameter passing between them.

TPMs are essentially specialized operating systems. Applications execute in the context provided by such environments. This makes it very difficult, if not impossible, to let applications of one TPM type invoke applications of another TPM type. WFMSs, on the other hand, are not like operating systems. Notions well known in the transaction processing area like backward recovery, atomicity, or compensation are non-existent in current WFMSs. TPMs provide some of these capabilities since they are especially designed for reliability, but they lack the notion of persistent forward execution, and the ability to coordinate the execution of related but independent activities. While the Workflow Management Coalition is proposing standard APIs for interoperability of WFMSs, no such standard APIs exist for TPMs.

In recent years, the deficiencies of the ACID transaction model have lead to the proposal of numerous *advanced transaction models* that extend or relax the ACID properties of classical transactions [9]. Common to most transaction models is the idea of supporting additional capabilities to control the interactions of groups of activities and provide relaxed ACID properties to groups of transactions instead of to individual transactions.

It is our belief that the existing ideas in the field of advanced transaction models will be implemented in WFMSs rather than in TPMs. The goals for the next generation of both WFMSs and TPMs are very similar. Since TPMs already support ACID transaction, the question arises whether TPMs are not better suited to meet the customers' requirements for advanced transaction models. However, it is more likely that, if advanced transaction models are finally going to make it as a viable commercial technology, this will happen in the workflow arena for, among others, the following reasons:

- WFMSs are designed to operate in modern information environments: distributed, client/server, heterogeneous, and with multiple applications.
- New and existing applications can be easily incorporated into a workflow environment without major modifications to either the workflow system or the application itself.
- WFMSs provide multiple key features that do not exist in other systems: coordination facilities, monitoring and auditing capabilities, abilities to reflect the organizational structure, and high level programming languages for the definition of processes.
- WFMSs incorporate users into the system. This is a first step towards a cooperative tool. Moreover, WFMSs provide support for defining individual users, grouping them into logical entities (roles) and assigning activities to particular users or roles.
- WFMSs reflect the organizational model and work environment in which they are used. They can be easily tailored for different environments.

However, with regard to advanced transaction models, the implementation of such models in WFMSs is not straightforward. There are many areas open to research. Some of the issues that need to be addressed are the following:

- Current WFMSs lack facilities to extend the ACID properties to groups of activities. The semantics of many operations such as recovery, rollback, compensation and undo are still undefined in WFMSs. There are many possibilities, but it is not trivial to evaluate their advantages and drawbacks.
- Legacy applications are here to stay. Better means must be devised to incorporate them in a workflow environment. As these applications need to be treated as blackbox applications, specialized *wrappers* are required to interoperate with existing applications.
- Although it is much used, the notion of *transactional workflows* [29] is still unclear. There is a wide gap between research advocating transactional workflow and what is being implemented in commercial systems. It is not clear, for instance, what are the requirements of WFMSs in terms of concurrency control and recovery. The *Workflow Management Coalition*, on the other hand, seems to be reluctant to incorporate the notion of transactional processing in its standards. This might be a problem of finding a common language.
- There is no single advanced transaction model that meets the requirements of all customers and applications. WFMSs will incorporate concepts rather than particular implementations: savepoints, nesting, coordination, compensation, invariants, dependencies, etc. and combine them into more flexible constructs. However, the exact semantics of such constructs are yet to be defined.
- There are many options to define the failure semantics for complex workflow processes. What should happen to a parallel branch if one branch fails? What should happen to the results of a sequence of activities which need to be reexecuted? Are they compensated for and in which order? How can a specific order of compensation steps be specified? All these questions need to be solved before solutions can be proposed.
- The workflow definition language is likely to end up providing a large set of primitives to specify transactional behavior. This set of primitives will be much richer, more complicated and clumsier than the ACID transaction primitives.

5 Conclusions and Future Work

The importance of WFMSs in large corporations cannot be stressed sufficiently. There is a high demand for such systems, but successful solutions will have to integrate many technologies and approaches. The challenge lies in providing systems capable of dealing with very large, heterogeneous, distributed and legacy applications, while providing an acceptable degree of reliability and availability. The research areas briefly discussed in this paper point towards some of the issues that need special attention in the design of WFMSs. These issues apply to the vast majority of existing systems, since they are not constraints imposed by a particular architecture but generic demands of the working environment targeted by these systems. In our case, we have focused on FlowMark since this is an IBM product and we have access to its code and developers. However, the solutions we proposed can be easily extended to other systems given that FlowMark's model (and architecture) closely resembles the generic model proposed as part of the Workflow Management Coalition standarization effort [13].

With the recent merger of IBM and Lotus, we have started exploring how to establish more synergy between Lotus Notes and FlowMark, taking advantage of the strengths of the two products. Some of the issues under consideration are: document-centric versus process-oriented workflow, structured versus ad hoc workflows, object orientation, external event handling, agent technologies, etc. Since the *send* model of workflow in Lotus Notes is similar in many ways to distributed workflows as we have discussed here, we have recognized its limitations based on our Exotica/FMQM work. We are in the process of doing further analyses.

Acknowledgements Part of this work has been done in collaboration with Amr El Abbadi and Divyakant Agrawal of the University of California at Santa Barbara while they were visiting IBM Almaden Research Center.

References

[1] G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan, R. Günthör, M. Kamath, *Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management*, **Proc. IFIP WG8.1 Working**

Conference on Information System Development for Decentralised Organizations, Trondheim, August 1995. Also available as **IBM Research Report RJ9912**, IBM Almaden Research Center, November 1994.

[2] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör, C. Mohan, *Failure Handling in Large Scale Workflow Management Systems*, **IBM Research Report RJ9913**, IBM Almaden Research Center, November 1994. See also http://www.almaden.ibm.com/cs/reports/workflow/exotica_papers.html

[3] G. Alonso, R. Günthör, M. Kamath, D. Agrawal, A. El Abbadi, C. Mohan, *Exotica/FMDC: Handling Disconnected Clients in a Workflow Management System*, **Proc. 3rd International Conference on Cooperative Information Systems**, Vienna, May 1995.

[4] Gustavo Alonso, Divy Agrawal, Amr El Abbadi, Mohan Kamath, Roger Günthör, C. Mohan, *Advanced Transaction Models in Workflow Contexts*, **IBM Research Report RJ9970**, IBM Almaden Research Center, July 1995. See also http://www.almaden.ibm.com/cs/reports/workflow/exotica_home_page.html

[5] A. Biliris, S. Dar, N. Gehani, H.V. Jagadish, K. Ramamritham, *ASSET: A System for Supporting Extended Transactions*, **Proc. ACM SIGMOD International Conference on Management of Data**, Minneapolis, May 1994, pp. 44-54.

[6] U. Dayal, H. Garcia-Molina, M. Hsu, B. Kao, M.-C. Shan, *Third Generation TP Monitors: A Database Challenge*, **Proc. ACM SIGMOD International Conference on Management of Data**, Washington, D.C., pp. 393–397, May 1993.

[7] A.K. Elmagarmid, Y. Leu, W. Litwin, M.E. Rusinkiewicz, *A Multidatabase Transaction Model for Interbase*, **Proc. 16th International Conference on Very Large Data Bases**, August 1990.

[8] D. Georgakopoulos, M. Hornick, A. Sheth, *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. **Distributed and Parallel Databases**, Vol. 3, No. 2, pages 119–153, April 1995.

[9] A.K. Elmagarmid (Ed.), *Transaction Models for Advanced Database Applications*, Morgan-Kaufmann, 1992.

[10] H. García-Molina, K. Salem, *Sagas*, **Proc. SIGMOD International Conference on Management of Data**, San Francisco, May 1987.

[11] H. García-Molina, D. Gawlick, J. Klein, K. Kleissner, K. Salem, *Coordinating Multi-transaction Activities*, **Proc. IEEE Spring Compcon**, San Francisco, 1991.

[12] C. Frye, *Move to Workflow Provokes Business Process Scrutiny*, **Software Magazine**, April 1994, pp. 77-89.

[13] D. Hollinsworth, *The Workflow Reference Model*, Workflow Management Coalition, TC00-1003, December 1994. Accessible via: http://www.aiai.ed.ac.uk/WfMC/

[14] *Message Queue Interface: Technical Reference*, IBM Document No. SC33-0850-01, April 1993.

[15] *FlowMark: Managing Your Workflow*, IBM Document No. SH19-8176-01, September 1994.

[16] *FlowMark: Programming Guide*, IBM Document No. SH19-8177-01, September 1994.

[17] T. Imielinski, B.R. Badrinath, *Mobile Wireless Computing: Solutions and Challenges in Data Management*, **Communications of the ACM**, Vol. 37, No. 10, October 1994.

[18] M. Kamath, G. Alonso, R. Günthör, C. Mohan, *Providing High Availability in Very Large Workflow Management Systems*, **Research Report RJ9967**, IBM Almaden Research Center, July 1995.

[19] F. Leymann, W. Altenhuber, *Managing Business Processes as an Information Resource*, **IBM Systems Journal**, Vol. 33, No. 2, pp. 326-348, 1994.

[20] F. Leymann, D. Roller, *Business Processes Management with FlowMark*, **Proc. 39th IEEE Computer Society International Conference (CompCon)**, February 1994, pp. 230-233.

[21] F. Leymann, Supporting Business Transactions Via Partial Backward Recovery in Workflow Management Systems, **Proc. GI-Fachtagung Datenbanken in Büro Technik und Wissenschaft - BTW'95**, Dresden, Germany, March 1995, Springer Verlag.

[22] S. Mehrotra, R. Rastogi, A. Silberschatz, H.F. Korth, *A Transaction Model for Multidatabase Systems*, **Proc. International Conference on Distributed Computing Systems**, June 1992, pp. 56-63.

[23] C. Mohan, R. Dievendorff, *Recent Work on Distributed Commit Protocols, and Recoverable Messaging and Queuing*, **Bulletin of the IEEE Technical Committee on Data Engineering**, Vol. 17, No. 1, March 1994, pp. 22-28.

[24] C. Mohan, *Advanced Transaction Models - Survey and Critique*, Tutorial presented at the **ACM SIGMOD International Conference on Management of Data**, May 1994. Available at http://www.almaden.ibm.com/cs/reports/workflow/tran_models_tutorial_sigmod94.ps.Z

[25] C. Mohan, K. Treiber, R. Obermarck, *Algorithms for the Management of Remote Backup Data Bases for Disaster Recovery*, **Proc. 9th International Conference on Data Engineering**, Vienna, April 1993.

[26] R. Obermack (Ed.), *Special Issue on TP Monitors and Distributed Transaction Management*, **Bulletin of the Technical Committee on Data Engineering**, Vol. 17, No. 1, March 1994.

[27] M. Hsu (Ed.), *Special Issue on Workflow Systems*, **Bulletin of the Technical Committee on Data Engineering**, Vol. 18, No. 1, March 1995.

[28] B. Reinwald, *Workflow Management*, Tutorial handout, **13th IFIP World Computer Congress**, August 1994.

[29] A. Sheth, Rusinkiewicz, *On Transactional Workflows*, **Bulletin of the Technical Committee on Data Engineering**, Vol. 16, No. 2, June 1993.

[30] B. Salzberg, D. Tombroff, *A Programming Tool to Support Long-Running Activities*, **Technical Report NU-CCS-94-10**, College of Computer Science, Northeastern University, Boston, 1994.

[31] A. Zhang, M. Nodine, B. Bhargava, O. Bukhres, *Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems*, **Proc. SIGMOD International Conference on Management of Data**, May 1994.