Transactional Business Process Servers: Definition and Requirements

Thomas Mikalsen, Isabelle Rouvellou, Stanley Sutton Jr., Stefan Tai

IBM T.J. Watson Research Center
Hawthorne
New York 10532, USA
{tommi, rouvellou, suttonsm, stai}
@us.ibm.com

Mandy Chessell, Catherine Griffin, David Vines

IBM United Kingdom Laboratories
Hursley Park, Winchester
England, SO21 2JN
{chessell, cgriffin, dvines}
@uk.ibm.com

ABSTRACT. *This paper discusses the implementation of transactional business processes using business object components. We define and formulate the idea of a transactional business process server (TBPS) and describe important requirements for TBPS systems. TBPS in combination with business components improve both the development and execution of automated transactional business processes.*

KEY WORDS: *Business process, business transactions, software component technology, workflow technology, process servers*

1. Introduction

Enterprise applications that must support complex business processes benefit to a great extent from business object component technology. Business objects are abstractions of both data and business process logic, allowing complicated access to legacy systems to be encapsulated and hidden from the enterprise application clients. Business processes are thus presented with an object model that is aligned with the enterprise domain model, but they do not have to be concerned with the details of rigid legacy systems and legacy architectures. This situation is depicted in Figure 1.

The current model has a significant limitation, however. That is, clients of the business objects must still include the programming of potentially complicated process control flow, transaction management logic, process recovery logic, and so on, which involve the business objects and any data access encapsulated by the business objects. This complicates the business process design and management, increases the opportunities for programming errors, and reduces application maintainability and extensibility.

Workflow management addresses some, but not all aspects of this problem (as discussed in Section 2), and it imposes abstractions and functionality that are not always appropriate for all business processes. Existing component technologies (for example, Enterprise JavaBeans (SUN, 1998; Monson-Haefel, 1999)) address some aspects of the problem, too, such as ease of programming complex transactions. However, they lack appropriate process abstractions.

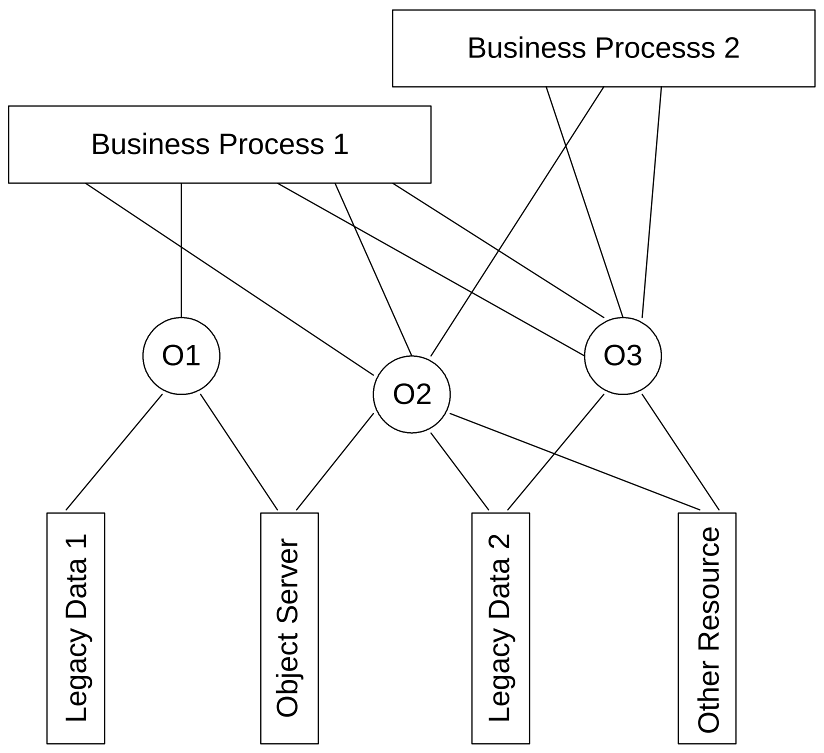

Figure 1. Business objects encapsulate legacy data systems, but business processes must still include complicated transactional access to business objects.

To address the problem we are proposing to enrich the services that are available to business processes for transactional use of enterprise data. Our considered approach is to encapsulate not only data but also the (transactional) logic of data use. In other words, while business objects and other resources should remain accessible as components, we also want to make significant patterns of use of business objects and resources accessible as components. In particular, we would make transactional business processes into server-side components available for use by distributed, client side business processes and applications.

This situation is illustrated in Figure 2. Here, instead of business object components, business processes involve *transactional business process components*. (Although not shown, direct access to business objects and even legacy systems may still be possible.)

Transactional business process components differ from common business objects in the level of process abstraction that they provide. The process abstraction offered by a business object, typically called a method, allows a client to request services from an object without knowing the details of how those services are provided. Such methods are typically synchronous and short lived. In other words, business object components offer subprogram abstractions.

When business objects are used and combined to support larger, more complex business processes, new patterns and commonality emerge: long-lived units of work, interruptible execution, fault-tolerance, process recovery, and so on. What is needed are new forms of process

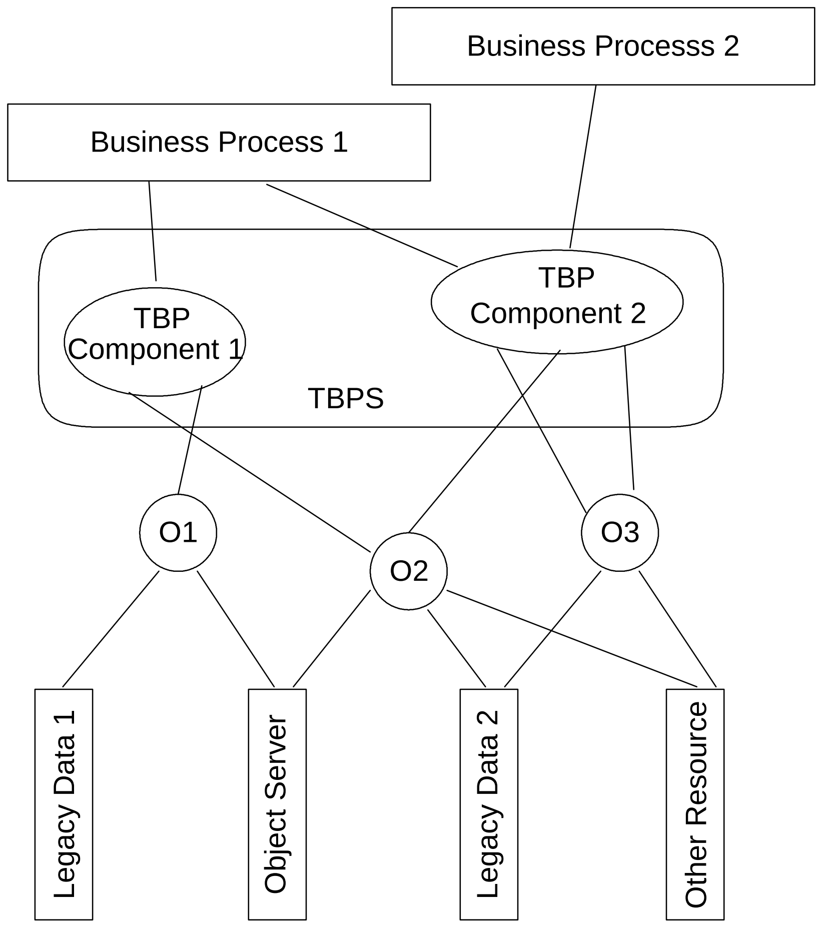

Figure 2. Transactional business process components encapsulate patterns of transactional access to business objects. Business processes can make simpler references ot these components, which can be reused across business processes.

abstraction, able to hide the implementation details required to support these common patterns. Such higher-level process abstraction is offered by transactional business process components.

Transactional business process components encapsulate the transactional access to business objects and data, hiding the complexity of that from the business process. Access to the transactional business process components themselves is relatively straightforward.

This approach is motivated by several prospective advantages:

- It enables separation of independent or evolving business logic from dependent or fixed transaction management (and related) data-use logic. This, in turn, simplifies business logic and business application programming. It also allows business logic to evolve without hindrance from transaction-related data-use logic, and conversely it allows transaction-related data-use logic to evolve without impacting independent business logic.

- The approach allows the separation of repeated patterns of transactional data use. These patterns would then be available for use in any type of business process or application, from often repeated, business-critical processes (“production processes”) to processes that are just *ad hoc* or “business-facilitating.” As components are repeatedly used, they can be validated and refined, thus raising the level of confidence in their correctness. The availability of validated components that represent established patterns of business data use facilitates the development of new business applications.

- The approach enables better integration of legacy data and resources in support of new and evolving enterprise applications. It provides not just the data abstraction but also a layer of process abstraction over legacy resources. This can be tailored to the purposes of new and emerging enterprise applications.

- Finally, this approach allows for improved process modeling for the programming of transactional uses of enterprise data and resources. The language used to program transactional business process components can be specialized for that purpose, designed especially to address the particular data-use requirements of business applications.

For all of these reasons, we believe that the use of transactional business process components running on transactional business process servers offers many potential economic and engineering advantages.

In Section 2 we discuss workflows and transactions to set the context for our definition of transactional business process servers. We present the definition of transactional business process servers in Section 3, and in Section 4 we give requirements for these servers and for modeling languages for transactional business process components. In Section 5 we discuss our plans for developing such a server, and Section 6 we present a summary.

2. Workflows and transactions

A careful definition and management of the business processes is of predominant importance in either of the two scenarios of common business objects (Figure 1) or transactional business process servers (Figure 2).

Over the past years, the use of workflow concepts and tools has been adverted for this purpose. A variety of commercial workflow systems are available, and workflow standards addressing the interoperability of proprietary workflow systems have been developed (WfMC, 1995; OMG, 1998c). At the same time, the relationship between workflow technology and business objects technology has emerged as an area of significant interest. Workflow components are deployed in a business component infrastructure which provides mechanisms for integrating business components that may be distributed, heterogeneous, and independently developed. Among the most important infrastructure mechanisms required is the support for transactions.
In this section, we briefly summarize important workflow and transaction concepts to set the stage for defining and proposing the concept of a *transactional business process server*.

2.1. Workflows

Important aspects of workflow are captured by standard terminology and architecture.

2.1.1. Terminology

Following the Workflow Management Coalition's (WfMC) basic terminology (WfMC, 1999), a *Business Process* is defined in a *Process Definition*, which is composed of *Manual Activities* and/or *Automated Activities*. The automated aspects of the business process are managed by a *Workflow Management System*, which controls *Process Instances*. Process instances are created from a process definition, and include one or more *Activity Instances*, which represent the process activities during execution. Activity instances include *Work Items* (tasks that are allocated to a workflow participant) and/or *Invoked Applications* (computer applications used to support an activity).

2.1.2. Architecture

The WfMC Reference Model defines the five components of *Process Definition Tool*, *Workflow Server*, *Workflow Client Application*, *Invoked Application*, and *Administration/Monitoring Tool*, and respective interfaces between these components.

Though the WfMC standard is an important step toward workflow system interoperability, the standard, as discussed in (Paul et al., 1997), introduces significant weaknesses impeding system flexibility and system scalability. This is mainly due to the centralized, monolithic nature of the workflow server component, which is responsible for multiple critical functions including process execution, distribution of work items, worklist management for workflow participants, and application invocation.

The WfMC standard demands users to introduce heavyweight workflow servers and modify architectures accordingly, which may not be necessary or appropriate for certain business processes and business component architectures. For example, a business process that is composed of automated activities only does not require the assignment and distribution of work items to workflow participants. Still, a sound process definition and process management may be highly desired or may even be critical. In such cases, rather than introducing a separate centralized workflow server, the business component itself (the application server) may be designed to offer selected process management functions. In other words, the business component becomes a *business process component*.

2.2. Transactions

As a workflow management system (or a business process server) coordinates the execution of various activities that constitute a single business process, the individual activities are not necessarily independent of each other, but may form a unit of work. The terms *transactional workflow* and *business transaction* are commonly used to refer to such business processes that have an "all-or-nothing" semantics (Leymann & Roller, 2000).

Business transactions are, however, different from *system-level transactions* as known from database systems, and as supported by middleware services like the CORBA Object Transaction Service (OTS). Business transactions must reflect the potentially *long duration* of business processes and must address both *transactional and non-transactional activities*.
Non-transactional activities are those activities that cannot automatically be undone but instead require some compensation mechanism.

While many concepts to support business transactions have been explored (such as the constructs of *atomic spheres* and *compensation spheres* to specify collections of activities as a unit of work and for backward recovery (Leymann & Roller, 2000)), only very few of the commercially available workflow management systems today support business transactions. This is partly due to the general difficulty of combining workflows and transactions, and to the limited availability of appropriate (system-level or higher-level) transaction services offered by the business component infrastructure into which the workflow components are to be deployed.

System-level transaction services, like the CORBA OTS or the Enterprise JavaBeans transaction model, do not meet the demands of business transactions, as such kinds of system-level transactions typically are short-lived and resource-locking transactions. In addition, they provide no activity compensation mechanism for non-transactional activities.

Alternative solutions like the OTSArjuna transaction service (Little & Shrivastava, 1999), and reliable workflow systems implemented on top of it (Wheather et al., 1998), provide a higher-level transaction toolkit that primarily aims to automate some transaction management aspects such as persistence or concurrency control. Still, support for business processes that are long-running and that include non-transactional activities is weak.

2.3. Towards transactional business process servers

We argue for process management support for transactional business processes that harmonizes well with modular and flexible business component architectures. We believe that workflow technologies and workflow architectures as suggested by the WfMC reference model have limitations because they are heavyweight and not well aligned with design principles of modern n-tier component architectures. We see the need for *transactional business process servers (TBPS)* that implement a superset of a subset of the functions of current workflow servers, to primarily address automated process definition and management on the (application) server-side.

3. Definition

A *Transactional Business Process Server (TBPS)* is an application server of one or more business process components supporting transactional business processes (business transactions).

Similarly to a workflow server, a TBPS manages the automated aspects of a business process that is described in a process definition. Differently from a workflow server, a TBPS excludes the workflow-automated aspects such as worklist management for workflow participants, assignment and distribution of work items, or invocations of user tools and applications that are in support of manual activities. A TBPS is “leaner” in that it is only concerned with activities that are carried out within the server. The activities may still involve external interaction with users, and a TBPS provides an interface and event-based mechanism for this purpose. A TBPS process may also invoke applications incidentally, but important computations are carried out within the TBPS.

Very importantly, a TBPS supports business transactions. The process activities are given a transactional semantics through an infrastructure service that supports long-running business processes and through a mechanism for process and activity compensation. The business process that a TBPS supports thus is a transactional unit-of-work. This unit-of-work may be part of a larger (workflow) process.

A TBPS includes a process definition tool and an administration tool that are in accordance with the above functions.

4. Requirements

Based on the problem we address, our general approach, and our particular motivations, we can elaborate a number of requirements for a transactional business process server and the components that represent transactional business processes. These requirements are described in the following subsections.

4.1 Transactional data use

Since a defining characteristic of TBPS is to support business processes in transactional use of business objects and data, the ability to process transactions across *various data stores* and other transactional resources in *distributed* environments is essential to a TBPS. A TBPS provides the (runtime) support for business objects representing resources which are to participate in diverse transaction protocols such as the CORBA OTS (OMG, 1998b) (and its Java binding JTS, for example), the X/Open DTP standard for distributed transaction processing (The Open Group, 1996), as well as advanced transaction protocols.

Ideally, TBPS should also allow "component-managed" access to data sources and resources as special cases. Some component data models, such as EJB, attempt to simplify this task by using declarative transactions (Monson-Haefel, 1999); however, such facilities are not part of the programming or process modeling language, and they are thus unnatural and potentially problematic to use.

4.2 Systems compatibility and interoperability

Since another defining characteristic of TBPS is to support the representation of transactional business processes as components, the processes should be representable as a component of some standard middleware technology such as CORBA (OMG, 1998a) or Enterprise Java Beans (EJBs) (SUN, 1998). Depending on the particular form of transactional business process components, support must also be provided for component life cycle and runtime management. In other words, the components must be operable within a standard component service environment and accessible via standard protocols for that environment. (WebSphere (IBM, 2000b) is an example of such an environment for EJBs.)

Regardless of how the transactional business process components are represented, they must be accessible by, and also able to access, typical enterprise applications and services. These include distributed objects such as CORBA objects (and object request brokers (ORBs)), EJBs, applications integrated through messaging middleware (such as MQSeries (IBM, 2000a)) workflow servers, web servers, and other business process components. The transactional business process components should offer and be able to execute both synchronous and asynchronous access.

4.3 Business process modeling and execution

A language for modeling transactional business processes must provide basic constructs for the representation for various kinds of activities, data, and resources.

To adequately represent the variety and dynamism of business processes, a flexible control model is required. The control model should allow activities to be executed concurrently and sequentially, conditionally and iteratively. It should allow activity instances to execute with or without synchronization, and it should allow activity instances to be created and destroyed dynamically. It should also support activity composition and decomposition.

With respect to data representation and flow, a transactional business process modeling language should provide for the representation of business objects and for values of appropriate base types. These must be adequate to capture references to external data objects or sources. Activities should be allowed data parameters and local data variables. The flows of data between activities must be represented, including normal and exceptional data flows. Issues related to the scoping and visibility of data in a business process should be addressed in a way that is consistent with the activity and control models of the language (and other language features, as appropriate).

Business processes do not operate entirely according to an imperative plan. They are subject to events, both good and bad. For the good events (or at least the events that are not definitely bad) some form of event handling is recommended. For the bad events, some exception handling mechanism is required. Ideally, both event mechanisms should be flexibly integrated with the process control model. Events can constitute one form of communication both within and between processes. Support for messaging can be provided as an alternative communication mechanism that may represent some forms of communication more explicitly. Access to an external messaging service is important for communicating business process messages to external processes and agencies.

For transactional business processes, naturally transactions should be an integral part of the process modeling language. Transactions in the process language should include at least flat, ACID transactions, both local and distributed. Support for advanced transaction models, including various kinds of nested transactions and long transactions (including sagas (Garcia-Molina & Salem, 1987) or the LRUOW transaction model (Bennett et al., 2000)) is highly desirable. The incorporation of advanced transaction models may entail the inclusion of associated features, such as compensation or versioning, that can further enhance process modeling capabilities. Constructs that are somewhat related to transactions, such as Component Broker sessions (which have the demarcation properties of transactions for resource management, but not the ACID properties) and security (IBM, 2000b) are also desirable and will also extend the flexibility and expressiveness of the language.

The need to support long running transactional processes implies some additional concerns for transaction management in a TBPS. Transaction models for TBPS must be sensitive to:

- Possible impediments to process concurrency, as might arise from the long-term locking of business objects or resources. Depending on the type of long-term transaction supported, this might entail nested transactions, flexible locking schemes, replication or versioning of data, and so on.

- Possible loss of work, for example, when a long-running process suffers an exception. This is more problematic for long transactions than for short transactions, since long transactions may involve more work that may be more difficult to redo. Possible approaches to mitigating the problem of loss of work include checkpointing, support for repair of problems that may cause failures, and support for commit in an inconsistent or incomplete state.

- The relationship of process structure to transaction structure. Both long-running processes and long-running transactions may be complexly structured. Providing flexibility in the relationship between process structures and transaction structures can allow process models to more naturally and adaptively match application domain structures. Some transaction management capabilities that are relevant to this concern include the ability to suspend and resume transactions, the ability to transfer transactions from one process and adopt them into another, and the ability to split and join transactions (Kaiser & Pu, 1992).

Compensation is another technique that may be used (with other mechanisms) to address some of the above concerns, such as concurrency in long transactions and flexibility in the relationship of process and transaction structures.

Naturally, the process modeling language must be automatically executable (or interpretable), according to a mechanism consistent with the component model and server-side technology adopted. However, it is also important that some provision be made for manual intervention in the execution of processes. Such intervention can be useful for resolving conflicts, handling errors and exceptions, adapting the direction of a process, and so on.

4.4 Business process adoption, reuse, adaptation, and extension

One of the chief advantages of component models is the ability to encapsulate useful data or functionality in a readily reusable form. One goal for TBPS is to leverage component technology to achieve this advantage for transactional business processes. Ideally, components in the process language should represent not just whole business processes (the coarsest level of granularity in process modeling) but also particular elements from which business processes are composed. These might include such things as activities, resources, data, and so on.

It should be possible to readily combine components to compose processes and, conversely, to decompose processes to extract generally useful subprocesses or subcomponents. It should also be possible to combine processes to create multi-process enterprise applications and to separate combined processes. These abilities can greatly facilitate business process (and process component) adaptation, extension, adoption, and reuse.

The above capabilities allow the tailoring of business processes through the modification of process models. It is also desirable to support the tailoring of business processes by providing mechanisms that allow adaptation within a given process model. Tailoring is possible at deployment time, at invocation time, and at runtime. Deployment-time tailoring is supported with EJB deployment descriptors (Monson-Haefel, 1999), for example. Invocation-time tailoring is supported by explicit invocation parameters and by mechanisms such as environment variables. Both of these mechanisms can also play a role at runtime. Another mechanism for runtime adaptation include dynamically adaptable rules, such as in Accessible Business Rules (Rouvellou et al., 2000). (Other mechanisms, such as dynamic allocation of enterprise resources (Podorozhny et al., 1998), enable runtime adaptation of a given process model, but this sort of mechanism may be best suited for more workflow-like process modeling and support systems.)

4.5 Development life cycle concerns

With respect to the life cycle of business process development and use, the main concern is to allow for the representation and inclusion of various stakeholders in the life cycle. In order to keep their respective concerns separate, and to accommodate their respective areas of expertise, the business process modeling language and system should support a life cycle in which various categories of stakeholders can adopt various suitable roles in process development, deployment, and use. The incorporation of developer roles into the development life cycle is part of the development model for EJBs. The incorporation of various stakeholders in various roles in the development cycle is also advocated for so-called "factored" process languages (Sutton & Osterweil, 1997).

5. Plans

We are working on the design and implementation of a transactional business process server, and our first results are very promising. Our TBPS has been implemented as a set of Java classes (JavaBeans) to run on IBM's WebSphere middleware, assuring portability and middleware connectivity to Enterprise JavaBeans as well as messaging systems (MQSeries).

A number of specialized services and tools have been developed to address the requirements identified above. These include a flexible graphical language and modeling tool for defining transactional business processes. With these, a process is expressed using process types (such as sequential process, parallel process, or cluster process), process patterns (for example, for activities accessing resources), and respective compensation processes and activities, if desired. The process modeling language includes hooks for using alternative transaction services for process elements that access business objects and resources.

The necessary transactional semantics for long-running transactional business processes is being provided through the *long running unit-of-work transaction model and service (LRUOW)* (Bennett et al., 2000). The LRUOW transaction model includes nesting with transactions that can be suspended and resumed by the same process or by different processes. Isolation between LRUOW transactions is provided by versioning, which facilitates concurrency. Versions are merged according to a predicate/transform protocol, which is generally automated, or a conflict detection and resolution protocol, which may include manual intervention.

6. Summary

Managing business processes is critical to organizations, and introducing effective and automated process management support is highly desirable.

Business object technology and workflow technology are promising approaches to this problem, but they have limitations because of the level of abstraction that they provide, and the functionalities and architectural constraints that they induce, respectively. Business objects offer procedural abstractions in terms of methods only; they do not represent complex process structures and patterns. Workflow technology typically imposes system architectures that center around a full-fledged workflow server combining process management and people/organization management. For many business processes, in particular those where functionalities such as distribution of work items and worklist management for participants are not needed, a workflow approach is too heavyweight and thus less applicable.

As an alternative, we propose the concept of a *transactional business process server (TBPS),* which we define as an application server of one or more *transactional business process* *components*. TBPS advance business object technology by providing a layer of process abstraction over business objects. Patterns of transactional use of business objects and other resources are made available as (reusable, composable) components, facilitating transactional process definition, management, and execution. A TBPS thus presents an alternative, leaner solution to workflow that is business object component-oriented.

We identify requirements for TBPS in the areas of transactional data use, systems capabilities and interoperability, business process modeling and execution, business process adoption, reuse, adaptation, and extension, and development life-cycle concerns. To address these requirements we are developing prototype tools and technology to support TBPS and TBPS systems. This will allow us to validate the TBPS concept and to identify and explore issues in TBPS system development and use.

References

B. Bennett, B. Hahm, A. Leff, T. Mikalsen, K. Rasmus, J. Rayfield, and I. Rouvellou. A Distributed Object Oriented Framework to Offer Transactional Support for Long Running Business Processes. In Joseph Sventek and Geoffrey Coulson (eds.), *Proceedings IFIP/ACM International Conference on Distributed Systems Platforms, New York , NY, USA April 2000 (Middleware 2000).* Lecture Notes in Computer Science 1795, Springer-Verlag, Berlin, Heidelberg, pp. 331-348, April 2000.

H. Garcia-Molina and K. Salem. Sagas. In U. Dayal and I. Traiger (eds.) *Proceedings of the 1987 SIGMOD International Conference on the Management of Data,* ACM, pp. 249-259, May, 1987.

IBM. MQSeries. http://www-4.ibm.com/software/ts/mqseries. 2000 (a)

IBM. Websphere. http://www-4.ibm.com/software/websphere. 2000 (b)

G. Kaiser and C. Pu. *Dynamic Restructuring of Transactions*. In A. Elmagarmid (ed.), *Database Transaction Models for Advanced Applications,* Morgan Kaufmann, San Mateo, California; Chapter 8, pp. 265-295, 1992.

R. Lewis. *Advanced Messaging Applications with MSMQ and MQSeries.* Que Corporation, Indianapolis, Indiana, 2000.

F. Leymann, D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall, 2000.

M. Little, S.K. Shrivastava. *OTSArjuna*. University of Newcastle upon Tyne, England. http://arjuna.ncl.ac.uk/OTSArjuna/index.html.

R. Monson-Haefel. *Enterprise Java Beans*. O'Reilly & Associates, Inc., Sebastopol, California, 1999.

OMG (Object Management Group). *The Common Object Request Broker,* 1998 (a).

OMG (Object Management Group). *CORBA Services,* 1998 (b).

OMG (Object Management Group). *OMG Workflow Management Facility*, Rev. Submission, 1998 (c); ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf.

The Open Group. *X/Open Distributed Transaction Processing Reference Model*, Version 3, February 1996; http://www.opengroup.org.

S. Paul, E. Park, J. Chaar. Essential Requirements for a Workflow Standard. *OOPSLA'97 Workshop on Business Object Design and Implementation III*, 1997.

R. Podorozhny, B. Staudt Lerner, and L. J. Osterweil. Modeling Resources for Activity Coordination and Scheduling. In *Proceedings of Coordination 1999, April 26-28, 1999, Amsterdam, The Netherlands*. Lecture Notes in Computer Science #1594, Springer Verlag, Berlin, Heidelberg, pp. 307-322, 1998.

I. Rouvellou, L. Degenaro, K. Rasmus, D. Ehnesbuske, and B. McKee. Extending Business Objects with Business Rules. In *Proceedings of the 33rd International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Europe 2000)*, Mont Saint-Michel/ St-Malo, France, Juin, 2000. IEEE Computer Society Press, pp. 238-249.

Sun Microsystems, *Enterprise JavaBeans*™ Specification, 1998.

S. Sutton, Jr. and L. J. Osterweil. The Design of a Next-Generation Process Language. In *Proceedings of the Joint 6th European Software Engineering Conference and the 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Springer-Verlag, Berlin, Heidelberg, pp 142-158, 1997.

S.M. Wheater, S.K. Shrivastava, F. Ranno. A CORBA Compliant Transactional Workflow System for Internet Applications. In *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98), The Lake District, England*, September 1998.

WfMC (Workflow Management Coalition). The Workflow Reference Model. Document-Number TC00-1003, Issue 1.1, January 1995. http://www.aiim.org/wfmc/mainframe.htm.

WfMC (Workflow Management Coalition). WfMC Terminoloy & Glossary. Document-Number WFMC-TC-1011, Issue 3.0, Feb. 1999.