

Transactions meet MOM – System Support for Integrating Distributed Object Transactions and Messaging in Java and MQ Environments

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

IBM T.J. Watson Research Center
30 Saw Mill River Road, Hawthorne, New York, USA
+1-914 784 7981

{stai | tommy | rouvellou | suttonsm}@us.ibm.com

ABSTRACT

The Dependency-Sphere project explores concepts and middleware system support for distributed transaction processing across object and messaging components. The objective is to enrich standard object middleware and messaging middleware to provide for an increased level of reliability for their use in combination in enterprise systems.

Keywords

Transactions, object middleware, messaging middleware, middleware integration, distributed systems

1. PROBLEM STATEMENT

The integration and interoperation of diverse enterprise applications (EAI) in a distributed, heterogeneous environment is of predominant importance in the development of enterprise systems. Over the past years, middleware technology has been introduced to facilitate EAI and has found widespread acceptance. Middleware is connectivity software that defines a set of services to mediate between various applications across different languages and platforms.

Many enterprise systems today employ both *object-oriented middleware (OOM)*, as exemplified by the OMG's CORBA and component technologies like Sun's Enterprise JavaBeans, as well as *message-oriented middleware (MOM)*, as exemplified by IBM's MQSeries and implementations of Sun's Java Message Service (JMS). Each kind of middleware has its advantages for specific EAI problems.

OOM, for example, provides support for *distributed object transactions* to guarantee atomic processing of a set of object requests. Therefore, object transactions, as defined with the CORBA OTS, J2EE JTS and EJB transaction models, are commonly used to address application reliability and correctness concerns. MOM, for example, supports *flexible messaging interaction models* based on asynchronous, middleware-mediated communication, which decouples components to preserve component autonomy and to tolerate partial failures.

OOM and MOM, though often required and desired within the

same architecture, have been developed independently of each other and their combined use is mostly undefined. MOM promotes decoupled, n-to-m, time-independent and asynchronous communication, whereas the standard object interaction model of OOM is reference-based, one-to-one, time-dependent and synchronous. MOM also supports a notion of a transaction, which, however, is fundamentally different from object transactions. Message-oriented transactions offer atomic message publication reliability for grouping the delivery of a set of messages into a single unit-of-work, but they do not integrate the execution of object requests. Conversely, distributed object transactions do not integrate the publication of messages that can affect the outcome of the object transaction.

The lack of a defined middleware support for *transaction processing across object and messaging components* is a major limitation for the development of enterprise systems. System engineers are forced to build their own integration solutions, which are ad hoc, very complex and difficult to develop and to maintain.

2. STRATEGIES FOR INTEGRATING DISTRIBUTED OBJECT TRANSACTIONS AND MESSAGING

The principal strategy for integrating distributed object transactions and messaging is to define a transaction programming model that allows to combine and to establish a mutual dependency between

- (synchronous) transactional object executions,
- (asynchronous) message deliveries to final recipients (beyond intermediary destinations like message queues), and
- (deferred synchronous) decoupled transactional processing of messages by final recipients.

Such an integration is often desired, for example, for the atomic creation of database entries in two different databases, where one database offers a synchronous EJB interface and the other an asynchronous MQSeries interface only. A structured and reliable integration solution that attains defined levels of quality-of-service (including support for recovery) is needed.

In the paper “*Strategies for Integrating Messaging and Distributed Object Transactions*” [1] we propose and discuss different levels for integrating messaging and object transactions. These range from including a message queue as a resource manager in an object transaction to integrating remote message deliveries and integrating the processing of messages concurrent to the ongoing object transaction.

3. MIDDLEWARE MEDIATED TRANSACTIONS

We propose *Middleware Mediated Transactions (MMT)* as a solution to the above problem. MMT are an evolutionary and integrative approach to realize advanced integration strategies by extending distributed object transactions to include messaging.

MMT allow the arbitrary mixing of standard transactional object requests with messaging operations in the same, single unit-of-work. MMT therefore combine high transactional reliability achieved with object transactions with weaker forms of reliability guaranteed for the more flexible interaction models of mediation-based messaging communication.

MMT overall define a distributed transaction model of “relaxed ACID” properties that can be implemented on top of conventional transactional object and messaging middleware. The paper “*Middleware Mediated Transactions*” [2] presents and discusses the general concepts and properties of MMT, abstracting from particular features of the previously introduced Dependency-Spheres system (see below) and another related system prototype.

4. DEPENDENCY-SPHERES

The *Dependency-Spheres (D-Spheres)* project at the IBM T.J. Watson Research Center in Hawthorne, New York, is an MMT approach based on distributed object and message queuing technology. The D-Sphere system focuses on enterprise systems that need to integrate J2EE applications with JMS/MQSeries applications.

As presented in the paper “*Dependency-Spheres: A Global Transaction Context for Distributed Objects and Messages*” [3], a D-Sphere defines a new type of a transaction context inside of which conventional object transactions and distributed messages may occur. The D-Sphere system is an advanced integration solution uniquely supporting features and functionality as sketched above; it is the first system to make object transactions and messages that are associated with application-defined conditions mutually dependent.

More specifically, among the features of the D-Sphere system are:

- a dedicated API to demarcate object requests of standard object transactions and a set of messages in a single unit-of-work,
- the ability to associate messages with application-defined conditions to determine messaging success (respective failure),
- the ability to send messages to remote destinations at any point in time during the course of an ongoing transaction and to observe message delivery and message processing by recipients,
- the ability to automatically determine an overall D-Sphere transaction outcome based on the individual outcomes of object executions and messages, and to take respective action of commit or recovery and compensation in case of success or failure.

The D-Sphere system has been implemented in Java using IBM’s WebSphere application server and IBM’s MQSeries messaging middleware. The system internally uses a set of persistent message queues (with reliable messaging and MOM transactions) for purposes of persistent logging, observation and evaluation of message condition satisfaction (respective violation), and compensation support. Details of the system architecture are described in [3].

All publications and up-to-date information related to the Dependency-Spheres project can be found on the Dependency-Spheres homepage [4].

5. REFERENCES

- [1] S. Tai, I. Rouvellou. “Strategies for Integrating Messaging and Distributed Object Transactions”, in *Proc. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000, New York, USA)*. Springer-Verlag LNCS 1795, 2000
- [2] C. Liebig, S. Tai. “Middleware-Mediated Transactions”, in *Proc. IEEE 3rd International Symposium on Distributed Objects and Applications (DOA 2001, Rome, Italy)*. IEEE Press, September 2001
- [3] S. Tai, T. Mikalsen, I. Rouvellou, S.M. Sutton Jr. “Dependency-Spheres: A Global Transaction Context for Distributed Objects and Messages”, in *Proc. IEEE 5th International Enterprise Distributed Object Computing Conference (EDOC 2001, Seattle, USA)*. IEEE Press, September 2001
- [4] S. Tai, T. Mikalsen, I. Rouvellou, S.M. Sutton Jr. “Conditional Messaging and Dependency-Spheres Homepage”. <http://www.research.ibm.com/AEM/d-spheres.html>