

Object Management Group object transaction service based on an X/Open and International Organization for Standardization open systems interconnection transaction processing kernel

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1997 Distrib. Syst. Engng. 4 151

(<http://iopscience.iop.org/0967-1846/4/3/004>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 38.107.179.210

The article was downloaded on 20/02/2012 at 20:52

Please note that [terms and conditions apply](#).

Object Management Group object transaction service based on an X/Open and International Organization for Standardization open systems interconnection transaction processing kernel

J Liang^{†‡||}, S Sédillot^{†¶} and B Traverson^{§+}

[†] INRIA, BP 105, F-78153 Le Chesnay, France

[‡] ENST, rue Barrault, F-75634 Paris Cedex 13, France

[§] EDF-DER, 1 avenue du Général de Gaulle, F-92141 Clamart, France

Abstract. This paper addresses federation of a transactional object standard—Object Management Group (OMG) object transaction service (OTS)—with the X/Open distributed transaction processing (DTP) model and International Organization for Standardization (ISO) open systems interconnection (OSI) transaction processing (TP) communication protocol. The two-phase commit propagation rules within a distributed transaction tree are similar in the X/Open, ISO and OMG models. Building an OTS on an OSI TP protocol machine is possible because the two specifications are somewhat complementary. OTS defines a set of external interfaces without specific internal protocol machine, while OSI TP specifies an internal protocol machine without any application programming interface. Given these observations, and having already implemented an X/Open two-phase commit transaction toolkit based on an OSI TP protocol machine, we analyse the feasibility of using this implementation as a transaction service provider for OMG interfaces. Based on the favourable result of this feasibility study, we are implementing an OTS compliant system, which, by initiating the extensibility and openness strengths of OSI TP, is able to provide interoperability between X/Open DTP and OMG OTS models.

1. Introduction

Today, increasing requirements on integrity and consistency of data resources mean that transaction processing (TP) plays an increasingly important role in information technology (IT). Based on networks, a complex application can span distributed computing systems. All these facts lead to the emergence of distributed transaction processing (DTP).

Oriented towards the object world, the Object Management Group (OMG) has specified an object transaction service (OTS) [1] that supports transactional behaviour in a distributed heterogeneous environment based on the common object request broker architecture (CORBA) [2]. The transaction services are defined in a set of interfaces, but the internal protocol machine (transaction service provider) is not specified.

^{||} E-mail address: Jian.Liang@inria.fr

[¶] E-mail address: Simone.Sedillot@inria.fr

⁺ E-mail address: Bruno.Traverson@der.edf.gdf.fr

Obviously, constructing from scratch a new transaction service provider with OMG interfaces is an expensive method if an existing one can easily be reused. For this reason, some transaction system vendors intend to extend their existing procedural transaction systems onto a CORBA architecture in order to benefit from the object oriented design (OOD) and object oriented programming (OOP). Two examples are Transarc's Encina++ and Hitachi's TPBroker.

However, for commercial reasons, most of such products look like black boxes whose internal mechanisms are difficult to adapt to users' specific or existing environment. In addition, each of them has its own product-dependent features, thus, the extension designs are different from one to another.

We believe that there are generic aspects in the design of transaction management system implementations which allow extension for the sake of interoperability between heterogeneous systems at a very reasonable cost. This paper describes an implementation process which proves that the

above statements are realistic.

In order to discuss such an extension in a popular, generic way, we need the term standard. A distributed transaction processing can be viewed as two major functionalities: transaction management and transaction propagation. Following this category, we address how to extend the existing DTP standard within the object world.

X/Open DTP model [3] is a DTP standard whose aim is to build compliant systems out of components provided by more than one software vendor, thus promoting open, flexible systems. The key component in this model is a transaction manager which provides global transaction management services. Taking an existing X/Open compliant transaction manager as a kernel for providing OTS transaction management interfaces is a meaningful attempt.

Officially part of the X/Open DTP model, a two-phase commit protocol—open systems interconnection distributed transaction processing protocol (OSI TP) [4]—is defined by the International Organization for Standardization (ISO). OSI TP does not define any application programming interfaces (API), and thus lets the protocol elements be easily used by new interfaces. Based on the complementarity with the OMG OTS model, the intention is to use OSI TP through OTS interfaces in order to support OMG's transaction protocol.

Moreover, OSI TP isolates the access to the lower layer network protocol, which opens the protocol machine to map onto different data structures. We also use this feature to access the object communication support.

INRIA has developed a running X/Open compliant transaction toolkit based on an OSI TP protocol machine named Open Atomic Actions transaction Manager (MAAO) [5], which provides both global transaction management and transaction propagation services. Therefore, the implementation described in this paper is based on the use of MAAO as a transaction service provider to build an OTS compliant transaction system named MAAO OTS.

The rest of the paper is organized as follows: sections 2 and 3 describe briefly and respectively the OMG OTS model and the MAAO architecture. The comparative analysis between OTS, X/Open and OSI TP standards is studied in section 4 in order to outline the basis of conceptual architecture. In section 5, we specify the architecture and the mechanisms of MAAO OTS. Section 6 describes the expected benefits of MAAO OTS. Section 7 briefly gives the related works. Section 8 describes the future works. Finally, section 9 concludes the paper.

2. OMG OTS

The OTS is one of the common object services defined by OMG and based on the CORBA architecture. The OTS architecture is shown in figure 1.

A transaction may involve multiple objects. The scope of a transaction is defined by a transaction context which is shared by participating objects. The transaction service provides transaction management services and transaction propagation protocol through a set of well defined interfaces. The object request broker (ORB)

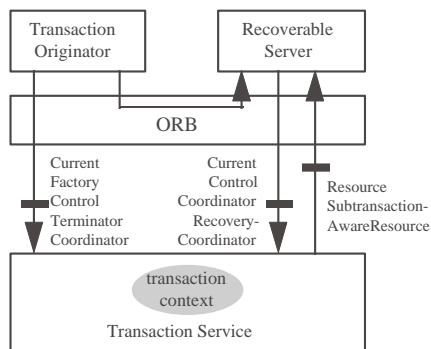


Figure 1. OMG OTS architecture.

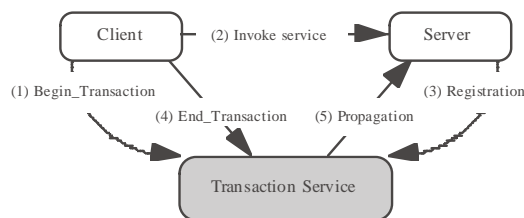


Figure 2. OMG OTS execution flows.

offers communication support for invoking operations and receiving the results transparently. The transaction originator is the client which creates a transaction and invokes application operations. A recoverable server, normally, consists of at least one application object and one resource object (RO).

A typical transaction is executed as shown in figure 2.

(1) A client invokes directly or indirectly a transaction service through the *factory* or the *current* interface in order to begin a transaction. The transaction service creates a transaction context and a globally unique transaction identifier.

(2) The client invokes the server including the transaction propagation context as an implicit or explicit parameter which contains the transaction identifier and the transaction service object reference.

(3) Upon receipt of an object request, the recoverable server registers its ROs with the transaction service through the *coordinator* interface.

(4) The client invokes the transaction service through the *terminator* interface to terminate the transaction (commit or rollback).

(5) The transaction service propagates the completion decision by sending two-phase commit [6] requests to the registered resources through *resource* interfaces.

The OTS specification defines a technique named interposition which allows multiple transaction services to cooperate in support of a global transaction. Interposition offers the interoperability between heterogeneous transaction services.

3. INRIA transaction toolkit

MAAO is an implementation of the transaction manager identified in the X/Open DTP model and based on the

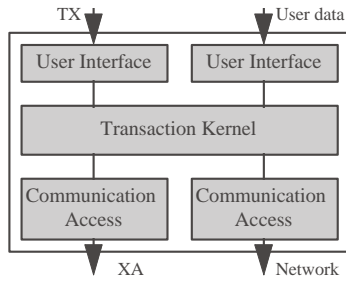


Figure 3. MAAO architecture.

design of the ISO OSI TP protocol machine. Including both the transaction manager and the OSI TP protocol, MAAO is an integrated transaction communication manager (ITCM) which provides both global transaction management services and communication protocols. MAAO architecture consists of three major components which are structured as three hierarchical levels as shown in figure 3.

The transaction kernel provides the OSI TP two-phase commit engine. this transaction kernel is an implementation of OSI TP multiple association control function's (MACF) services and protocol. It provides transactional services to the user interface through OSI TP specified high-level functions and uses communication access through OSI TP specified low-level functions. The transaction kernel is isolated from execution environment, physical distribution, system and network by its up-layer user interface and low-layer communication access.

The user interface layer can be customized to support various application programming interfaces. Currently, the user interface is customized to offer both the X/Open TX interface [7] in order to support transaction delineation services and a communication service (send/receive) which allows applications to communicate with each other.

The communication access layer provides the transaction kernel with transparent access to communication system (may be middleware, local communication toolkit or network access). The communication access implementation is based on the OSI TP single association object's (SAO) concept according to the specific context maintained for each transaction branch flow (state, encoding rules, transport stack, data transfer paradigm). A MAAO transaction branch can be either an application-to-application (client/server) branch, or a transaction manager to resource manager (RM) (Database) branch. In this case, X/Open XA interface [8] is substituted to the transport stack. This has been made feasible by observing that the transaction completion flows are similar on the two types of branches.

Based on the ISO OSI TP, MAAO has two important characteristics or strengths compared with other transaction systems.

Openness. The aim of X/Open standards is to promote meaningful open systems. ISO OSI TP is a standard whose purpose is to allow transaction systems to interoperate. MAAO, as an implementation of both may benefit from the open concept which is very important for the interoperability.

Extensibility. MAAO's transaction kernel is independent of the user interface and underlying communication

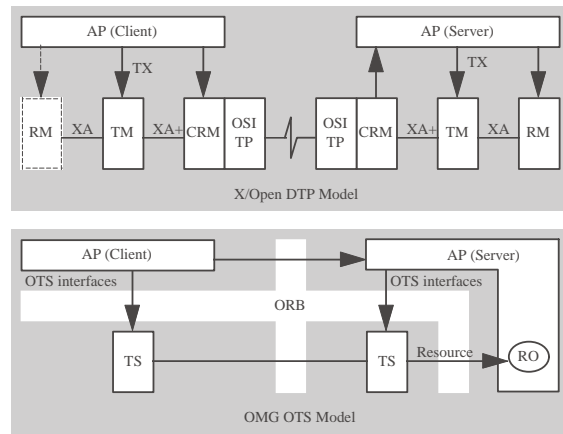


Figure 4. Model comparison.

support. This independence is very useful for following works where all the conceptions and developments may take place at the user interface and communication access layers without changing the transaction kernel. In other words, MAAO is a system which can be easily extended by various interfaces and communication supports to build distinct distributed transaction processing systems.

4. Comparative analysis

Our objective is to take existing X/Open and OSI TP implementations to act as the transaction kernel for OTS specified interfaces. The constraint of this transaction service provider is that it should be rich enough to support an OMG specified interface, and it should be an extensible system to be easily integrated with others. This transaction kernel should offer two main functionalities: global transaction management and transaction propagation. In this section, we compare the X/Open, OSI TP and OMG OTS from these two perspectives to analyse the feasibility of this approach.

4.1. Model comparison

Both X/Open DTP and OMG OTS models consist of a set of fundamental components. The model comparison attempts to identify the corresponding functional components and relevant interfaces in the two models. Figure 4 illustrates the two models.

Data accessed by an application (AP) are encapsulated in a RM according to the X/Open DTP model, and in a RO in the OTS model. In the OTS model, a client application always accesses a RO through an operation invocation on a recoverable server like that of remote RM access in the X/Open DTP model. In particular, the X/Open DTP model represents the client application as accessing directly local RMs. However, none of the current implementations (Tuxedo, Encina, etc) approaching the X/Open DTP model follow this design. All the data accesses are performed by the server. Therefore, we can declare that client and server applications/objects have analogous distribution and roles in both models.

Table 1. Equivalent components in X/Open and OTS.

X/Open DTP	OMG OTS
Application	Application object
Transaction manager	Transaction service
Resource manager	Resource object
Communication resource manager	Transaction service + ORB

Table 2. Equivalent interfaces in X/Open and OTS.

X/Open DTP	OTS Interface
TX	Current (factory, terminator, coordinator)
XA	Resource
OSI TP messages	Resource
XA+	No equivalent

A global transaction may involve multiple applications and resources. In order to ensure the ACID properties, the transaction manager provides essential functionalities in order to:

- (1) control the scope and duration of a transaction;
- (2) allow multiple resource to be involved in a single, atomic transaction, and to associate changes in their internal state with a transaction,
- (3) coordinate the completion of transactions on all participating resources;
- (4) drive failure recovery.

Function 1 (transaction begin, commit and rollback) is typically supported by the X/Open TX interface and by the OMG current or factory, terminator interfaces. Functions 2–4 (transaction branch begin, prepare, commit and rollback) are supported by the X/Open XA interface and by the OMG *resource* interface.

Functionally, all these tasks are fulfilled by the transaction manager in the X/Open DTP model and transaction service in the OMG OTS model. Therefore, in terms of providing global transaction management services, an OMG transaction service is equivalent to an X/Open transaction manager. A resource involved in a global transaction should:

- (1) provide an appropriate native interface which allows an application to manipulate data;
- (2) be coordinated as a direct subordinate to a transaction manager;
- (3) be registered with a transaction manager, statically or dynamically in order to participate in a transaction;
- (4) support an atomic transaction protocol to ensure the atomicity in a transaction;
- (5) have a concurrency control in order to participate in transaction isolation.

All these functionalities are provided by the RO in the OMG OTS model and the RM in the X/Open model. Thus, an OMG RO is equivalent to an X/Open RM.

In a DTP system, there are two requirements for communication: on one hand, the client/server requires an application-to-application communication paradigm, and on the other hand, coordinating the resources at the tree of the transaction termination requires pure transactional

semantics such as prepare, commit, rollback to be propagated. Thereafter, we call transaction propagation for all the communication protocol elements related to this second requirements.

In the X/Open DTP model, both the application-to-application communications and the transaction propagation are under the control of communication RMs which interoperate through the OSI TP protocol. The XA+ [9] interface between the transaction manager and communication RM is used in order to relay the transaction semantics between a transaction manager and the network.

In the OMG OTS model, both applications and transaction services invoke interfaces through the ORB, whether the services are local or remote. Communication, if it occurs, is transparent.

Therefore, we conclude that under the respect of transaction propagation, the role of the OTS and ORB are equivalent to that of a transaction manager and its relation to a communication RM. Precisely, an OTS with the ORB is equivalent to a transaction manager integrated with a communication RM (the XA+ interface is not used), called ITCM in the X/Open DTP model. Tables 1 and 2 summarize the equivalent components and relevant interfaces in the two models.

The model comparison results as follows:

Result 1. An X/Open transaction manager may act as the kernel of OMG transaction service to support OMG specified interfaces for transaction management.

Result 2. The OMG transaction service, implying the use of the ORB, can be considered as an ITCM which provides both transaction management services and transaction propagation protocol.

4.2. Transaction propagation comparison

The transaction propagation comparison between ISO OSI TP and OMG OTS aims at identifying the common characteristics of these two protocols.

In order to ensure transaction semantics to be reliably propagated between distributed participants, a superior/subordinate relationship is defined. All superior/subordinate relationships in a global transaction form a hierarchical transaction tree.

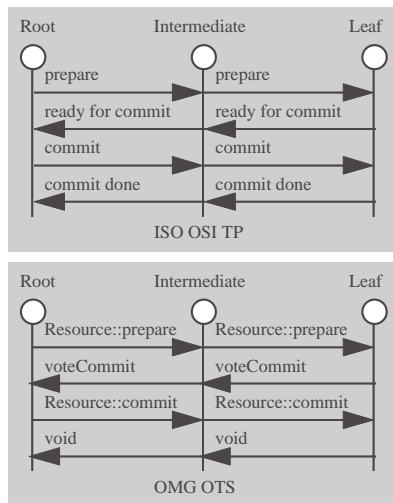


Figure 5. Transaction protocol comparison.

Corresponding to this necessity, in the OSI TP kernel, the relationship of TP protocol machines forms a distributed transaction tree in which the superiors and the subordinates communicate using OSI TP specified messages. In the OMG OTS, the relationship between involved transaction services forms a distributed transaction tree in which the communication is provided transparently by the ORB upon the occurrence of an object request on *resource* interface.

The rule of transaction propagation are identical: a superior may have one or more subordinate(s), but a subordinate can have only one superior. A subordinate may also, if required, become a superior of other subordinate(s). Each subordinate communicates only with its superior, there is no direct communication between the subordinates of the same superior. Thus, the concepts of distributed transaction tree in the OSI TP and in the OMG OTS are identical. Figure 5 shows the successful transaction completion propagated in the transaction tree in which a root node has no superior; an intermediate node has a superior and one or more subordinates; a leaf node has a superior and no subordinate.

Comparing transaction propagation in the transaction tree, the results that we obtain are that:

- (1) in both protocols, the transaction completion is always started and coordinated by the root;
- (2) both ISO OSI TP and OMG OTS support the two-phase commit protocol;
- (3) the sequences of information exchanges are identical in both protocols;
- (4) the message semantics in both protocols are similar.

We also compared other mechanisms such as rollback and heuristic and the analogous results that were obtained. Indeed, through a detailed comparison, we found that ISO OSI TP is a protocol which is richer than OMG OTS. Some of its functionalities do not appear in the OMG. For example, OSI TP allows us to prepare a given dialogue without preparing others. Therefore, we also consider that the OTS services and protocol are a subset of those specified by the ISO OSI TP. In other words, all the OMG OTS services (interfaces and functionalities) can be supported

by the OSI TP. Consequently, the transaction protocol comparison results as follows.

Result 3. OSI TP's transactional services and protocol are rich enough to support OMG transaction service for transaction propagation.

4.3. Conclusion of comparative analysis

Having compared the OMG OTS, X/Open DTP and ISO OSI TP from the model and from the transaction protocol points of view, we conclude the following statements in the form of logical syntax in which *r* = result; *c* = condition; and *d* = deduction:

$$r1 \wedge c1 = d1$$

$$r3 \wedge c3 = d3$$

$$r2 \wedge c2 = d2$$

$$d1 \vee d2 \vee d3 = \text{conclusion}$$

where

r1 = an X/Open transaction manager may act as the kernel of OMG transaction service in order to support the OTS specified interfaces concerned with transaction management;

r2 = an OTS transaction service is considered as an ITCM to support both transaction management and transaction propagation;

r3 = OSI TP's transactional services and protocol are rich enough to support OMG transaction service for transaction propagation;

c1 = MAAO supports the behaviour of the X/Open transaction manager;

c2 = MAAO is an implementation of ITCM;

c3 = MAAO implements OSI TP services and protocol;

d1 = MAAO may act as the kernel of the OMG transaction service in order to support OMG specified interfaces concerned with transaction management;

d2 = MAAO may satisfy the OTS's requirement on ITCM aspect;

d3 = MAAO may support OMG transaction service for transaction propagation.

Conclusion = MAAO may act as the kernel of the OMG transaction service.

Result 1, obtained from the model comparison, shows that an X/Open transaction manager may act as the kernel of the OMG transaction service to support OMG specified interfaces concerned with transaction management. Since the MAAO supports X/Open transaction manager's behaviours, we deduce that MAAO may act as the kernel of the OMG transaction service to support the OMG specified interface concerned with transaction management (*d1*).

Result 2, obtained from the model comparison, indicates that the OMG transaction service can be considered as an ITCM which provides both transaction management and transaction propagation within a unique component. Since MAAO is an ITCM, MAAO may satisfy OMG transaction service's requirement on ITCM aspect (*d2*).

Result 3, obtained from transaction protocol comparison, demonstrates that the transaction propagation protocol

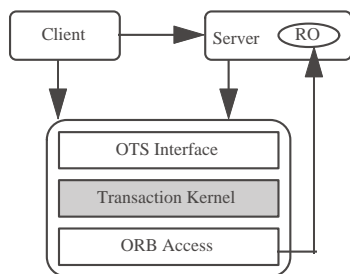


Figure 6. MAAO OTS architecture.

specified by the OMG can be supported by the ISO OSI TP. This means that OSI TP may act as the kernel of the OMG transaction service to support transaction propagation. Since the MAAO kernel is an implementation of the OSI TP protocol machine, the deduction from result 3 is that the MAAO may act as the kernel of transaction service for transaction propagation (*d3*).

From these deductions, we found that the MAAO may act as the kernel of the OMG transaction service to build an OTS compliant system, therefore called MAAO OTS.

5. MAAO OTS conceptual architecture

In the MAAO architecture, the user interface and communication access can be considered as the interfaces to the external world (execution environment, physical distribution, communication system, etc) for the transaction kernel and may be specialized to support distinct APIs and data transfer paradigms respectively. The transaction kernel (OSI TP protocol machine) provides transaction management services and a transaction propagation protocol which are independent of the external world. Thus, designing MAAO OTS is restricted to the both mappings in the interfaces and the ORB access.

In our architecture, the MAAO is extended onto CORBA architecture to act as an OMG's transaction service. In order to provide object environment and respect OSI TP characteristics, a new user interface is customized to provide OTS defined interfaces. Being transparent to object concept, the transaction kernel is maintained unchanged and provides two phase commit services which are used for both transaction management and transaction propagation. A new type of communication access is customized to provide access to ORB (see figure 6).

5.1. Relationship establishment

When a global transaction involves multiple participants, the superior/subordinate relationship should be established to form a transaction tree. This superior/subordinate relationship is used to ensure transaction propagation during the two phase commit process.

The OSI TP standard specifies such a relationship as a transactional dialogue based on a connection-oriented protocol (normally based on an OSI transport protocol). The nature of the OSI TP mechanism is that both user data

and transaction semantics have to be transmitted onto the same connection, thus on a single data flow.

The dialogue is established previously to forward user data and transaction semantic. Before sending user data, the superior already has the knowledge of its subordinates. So, establishing the superior/subordinate relationship is always required by the superior.

The nature of the OTS mechanism is that, user data and transaction semantics are transmitted independently in different flows. Client and server communicate directly, the transaction semantics are propagated between transaction service and RO.

Before sending its user data, the transaction superior application obtains from the OTS all information about its transaction and transmits it to the server. The server has to dynamically register its RO with the superior OTS.

The user interface provides an OTS specified *coordinator* interface to receive the registration request. The *coordinator* interface maps registration onto the transaction kernel in order to establish a logical, abstract dialogue to the subordinate. The ORB access remembers the object reference of the registered RO. The dialogue establishment does not result in any outgoing message but in the memorization of object reference of RO. Thus, the transaction kernel will be able to propagate transaction semantics (prepare, commit, rollback, etc) through ORB access to the registered ROs.

Interposition allows heterogeneous OTS compliant systems to interoperate in a global transaction. OSI TP specifies the way to initiate a subordinate protocol machine to a superior one, so MAAO OTS may be easily initiated as a subordinate to another OTS implementation. The creation of MAAO OTS as a subordinate results in an ORB access and an instance of transaction kernel being created. The MAAO OTS registers as a RO with superior. The *resource* interface is offered by the ORB access for receiving transaction completion requests.

It is worth noting that, in the OTS specification, there is no appropriate operation in the *factory* interface to support interposition, so we propose to add an additional operation join (propagation context) in the *factory* interface to clearly specify the creation of subordinate transaction service.

5.2. Database access

In the OTS specification, the RO manages the data resources involved in a transaction. Unfortunately, no existing database currently supports the *resource* interface. As a matter of fact, many existing relational database products such as Oracle, Informix, Sybase, DB2, etc support the X/Open XA interface. Some object database products, such as O2, also offer the XA interface. Consequently, we have also investigated the way to make the transaction service interwork with existing XA compliant databases.

Based on the analysis that the OMG *resource* interface corresponds to the X/Open XA interface, the current realization of the work is to let RO play an intermediate role between transaction service and database which maps *resource* interface onto the XA interface, as shown in figure 7.



Figure 7. Use of existing database.

During the initialization of a server, the `xa_open` is called to open the XA compliant database. Opening the database implies that a relationship between the recoverable server and the database is created. This relationship is represented by a thread of control. Closing of the database is performed when the recoverable server is shutdown.

An application object can be invoked by multiple clients and therefore be involved in multiple transactions. In order to optimize the utilization of the database, a thread of control should not be occupied by a transaction when the latter has no current manipulation on data. For example, if a transaction accesses twice with a database, the thread of control may be dissociated from the transaction between the end of the first intervention and the beginning of the second one, that allows the other transactions to associate with the thread of control in this interval. The XA compliant database may be dynamically involved in a transaction, the duration of involvement is enclosed by a pair of calls: `xa_start` and `xa_end`.

The database is involved in a transaction when the first request arrives at the server. This request causes a `xa_start(TMNOFLAGS)` to be called. Returning result of object request implies a `xa_end(TMSUSPEND)` to be invoked that the current transaction is temporarily dissociated from the database. The subsequent request belonging to the same transaction resumes the thread of control to associate once again with the database.

In the transaction completion process, when the RO receives the prepare request from the transaction service, it first ends definitively the association of the thread with the transaction by calling `xa_end(TMSUCCESS)`. Then it issues `xa_prepare` to begin the two-phase commit.

Since the application object and RO execute in the same process, they can share the same transaction context and thread of control. In this case, the database can guarantee that manipulation on data and the transaction completion requests belong to the same transaction.

The scenario of a transaction interacting twice with a database with successful completion is illustrated in figure 8. This figure demonstrates the MAAO OTS interworking with a X/Open XA compliant database.

In figure 8, the shaded parts signify the occupation of the thread of control, the other transactions may be associated with the thread in the white intervals to access the database.

5.3. Parallelism

Since a transaction service may support multiple transactions, and each of these transactions involves multiple applications and resources, the performance of an OTS system has to be seriously addressed.

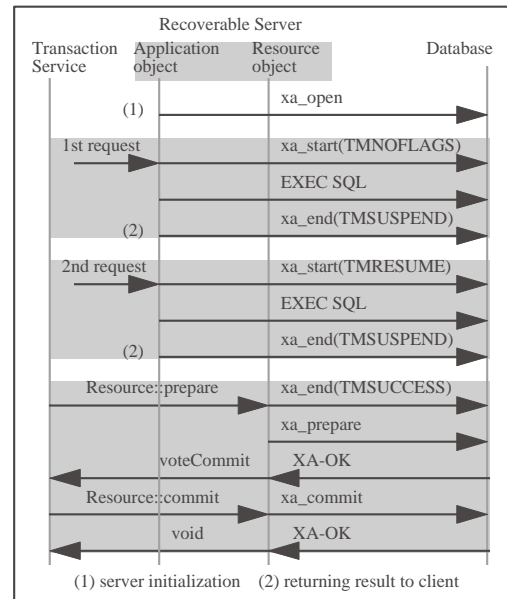


Figure 8. Database access.

Using the IONA Orbix 2.0 [10, 11] multiple threading (MT) package, we implement MAAO OTS with multiple threads in a hierarchical structure.

The transaction factory object is an object manufacturer which waits for solely transaction demarcation requests. When it receives a begin transaction request, it creates an instance of MAAO OTS which supports a given transaction. The transaction factory object runs in an individual thread and waits for the transaction demarcation requests at all times.

In order to improve the performance, we implement every instance of MAAO OTS within a thread. Thus, if there are m transactions at a given time, there would be m threads to perform these transactions in parallel.

During the two-phase commit process, the MAAO OTS sends transaction completion requests to all its resources. In order to avoid being blocked from waiting for the response, we create one thread for each registered resource. Thus, if there are n resources involved in a transaction, there would be n threads.

Totally, at a given time, if we suppose there are m transactions and in each transaction there are n involved resources, the total number of threads would be $1+m+m*n$.

6. Expected benefits

6.1. X/Open and OTS application interoperability

OMG OTS describes the model interoperability between the X/Open DTP domain and the OMG OTS domain. However, the description is too vague and not detailed enough to be easily understood. It seems that a transaction manager provides both X/Open and OTS interfaces to support the different types of applications. Such interoperability is only at application level—the different applications share the same transaction manager via different interfaces.

MAAO, by benefiting from OSI TP extensibility strength, allows the developer to extend various APIs and communication accesses without modifying the transaction kernel.

At user interfaces level, the multiple APIs may coexist to serve different applications. With this facility, in a single, global transaction, MAAO can support X/Open procedure applications via its X/Open interfaces and support OMG applications through its OTS interfaces, therefore, providing interoperability between heterogeneous applications.

At the communication access level, the different resource accesses may also coexist to interact with the database via different interfaces. Therefore, MAAO is able to manage the database both through the OMG's *resource* interface and directly via the X/Open XA interface.

6.2. X/Open transaction manager and OMG transaction service interoperability

The interoperability between heterogeneous transaction domains does not appear in the OTS specification. Such interoperability allows an application developed in a transaction domain to interoperate with an application developed in a different heterogeneous transaction domain. A domain is defined as containing its specific transaction management and transaction propagation protocol.

MAAO OTS must be looked at as an 'objectized' MAAO. We have shown that MAAO offers the X/Open DTP transaction management and the OSI TP-based transaction propagation. MAAO OTS adapts both of these with OTS interfaces and ORB access, which does not exclude further configuration.

OSI TP is part of the X/Open DTP model. Both X/Open DTP and MAAO OTS use OSI TP as transaction propagation protocol, so these two heterogeneous transaction manager domains may interoperate based on the same standard transaction protocol (see figure 9.1).

Of course, the communication paradigms of these two models are different: X/Open uses original OSI TP communication protocol with specified messages transmitted on the network; while the MAAO OTS communicates across ORB in the form of object request. To resolve such heterogeneity, we have to develop a communication box which translates OSI TP messages to CORBA object requests and vice versa. It is worth noting that this development concerns only communication forms, not transaction semantics.

On the other hand, by supporting the interposition technique, MAAO OTS can easily interoperate with another OTS (figure 9.2).

Because the MAAO X/Open DTP interface can coexist with the MAAO OTS interface on the same transaction management instantiation, it is possible to have a MAAO instantiation acting as a bridge between an X/Open domain and an OTS domain (see figure 9.3). This would allow a CORBA transactional application to span on an X/Open DTP domain. However, the application-to-application communication paradigm must be chosen among the three X/Open standards (TxRPC, XATMI and CPIC) and the

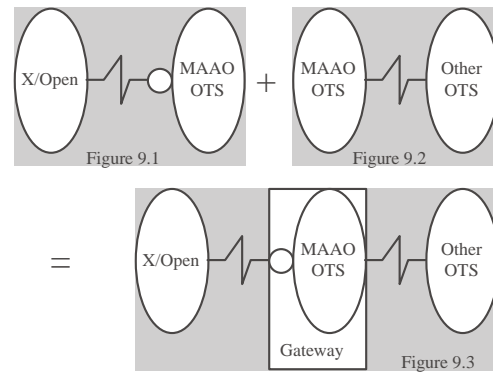


Figure 9. Interoperability between X/Open domain and OMG domain.

mapping between client request in one domain onto service from the other domain must be solved. This objective is part of the CEE advanced communication technologies and services (ACTS-ACTranS) project for the transactional DCE remote procedure call option (TxRPC) [12].

This transactional gateway enables the transaction propagation semantics to be relayed from one transaction domain to another. In each domain, the applications work uniquely with their own transaction manager, and ignore completely the type of transaction system in the other domain.

Acting as a transactional gateway between heterogeneous transaction managers in the X/Open DTP and the OMG OTS models, the MAAO OTS demonstrates its openness and standardization strengths compared with other OTS compliant systems. These strengths are deduced from that of ISO OSI TP.

7. Related works

Today, in order to benefit from the object technology concept, many transaction system vendors intend to build OMG OTS compliant transaction systems. It is necessary to present the current situation with related works.

Having the same objective as ours, Transarc is trying to build an OMG OTS compliant system based on its famous, industrially widely accepted transaction system—Encina. Such an object transaction system is named Encina++ [13] which extends existing Encina within the object world. Transarc's interest is to integrate its DCE and Encina products with DCE-based ORBs.

Hitachi is extending its X/Open DTP compliant distributed transaction processing system—Open TP1 [14] onto PostModern/Visigenic's CORBA 2.0 compliant ORB—ORBeline [15] that builds an OMG OTS compliant transaction system named TPBroker [16].

The Bull group is building its OTS system [17] onto IONA Orbix 2.0 MT with integrating Oracle RDBMS.

HP implemented its own ORB in Smalltalk with a transaction service named Distributed Smalltalk implementation on Transaction (DST) [18].

8. Future works

8.1. Subtransaction unavailability

The subtransaction concept provides a finer granularity of failure handling. The availability of subtransaction has recently been required more and more. However, to date, the ISO OSI TP cannot provide such flexibility yet, so MAAO, as an implementation of OSI TP, does not have subtransaction semantics. For this reason, the subtransaction does not get taken into account in the current version. Only the flat transaction is supported. Extending OSI TP with subtransaction semantics is a work in progress performed by ISO and will be available in the near future. The MAAO OTS may be enhanced when the subtransaction protocol becomes available in the OSI TP.

8.2. Object distribution

The set of OTS interfaces can be implemented by one or more CORBA objects freely. If multiple objects cooperate to constitute a transaction service, they are able to be extended on more than one host and communicate with each other across ORB. This distribution demonstrates the flexibility of extending the transaction service to a distributed system.

Due to the lack of some internal interface between transaction service objects such as interface between terminator and coordinator, and in order to facilitate the management of transaction context and thread of control, such as HP DST and Bull OTS, we implement multiple interfaces on an object. The object distribution is considered as a future improvement.

9. Conclusion

There is some complication in taking an X/Open procedural transaction manager whose kernel is an OSI TP protocol machine and making it into CORBA architecture. There is a technical challenge in designing a transaction manager for interoperability between heterogeneous transaction management systems. Based on the comparison of the transaction management functionalities in OSI TP, X/Open and OTS, we have proved that it is possible to design an open kernel with interfaces that can be adapted to either a procedural or an object oriented application.

The OTS interfaces encapsulate OSI TP protocol machine in order to make the architecture conformant with the OMG OTS standard. The interoperability with other OTS compliant implementation is provided by supporting interposition technique.

With the benefit of ISO OSI TP's extensibility strength, MAAO can easily be extended within the object world without reducing its existing functionalities, so that it may provide various interfaces (X/Open and OTS interfaces) to different applications. This facility allows the heterogeneous applications to coexist and interoperate through MAAO. MAAO can be also customized as a

transactional gateway to support interoperability between X/Open transaction manager and OTS domains.

The first conclusion of our experience is that the OTS interfaces and propagation specifications are simpler than those of X/Open, partly due to the use of ORB which makes communication transparent, partly due to the benefit of having been specified after X/Open DTP. The second conclusion is that the OSI TP propagation model features are rich and can support different transaction management features, among which those of an X/Open transaction manager or of OTS.

Waiting for the availability of subtransaction in OSI TP, the future work for MAAO OTS is to enhance current architecture with subtransaction semantics and to achieve object distribution.

References

- [1] OMG 1995 *Object Transaction Service. CORBA services: Common Object Services Specification (OMG document 95.3.31)* ch 10
- [2] CORBA 1995 *Common Object Request Broker Architecture: Specification and Architecture* revision 2.0
- [3] X/Open guide 1993 *Distributed Transaction Processing. Reference Model* version 2, G307
- [4] ISO/IEC 10026 Information Technology 1992 *Distributed Transaction Processing. Part 1: Model, Part 2: Service Definition, Part 3: Protocol Specification*
- [5] MAAO Toolkit 1995 *Kernel Programmer's Manual* version 1.0
- [6] Gray J 1978 *Operating Systems—An Advanced Course: 5. Transaction Management (Lecture Notes in Computer Science 60)* (Berlin: Springer)
- [7] X/Open CAE Specification 1995 *Distributed Transaction Processing: The TX (Transaction Demarcation) Specification C209*
- [8] X/Open CAE Specification 1992 *Distributed Transaction Processing: The XA Specification C193*
- [9] X/Open Snapshot 1994 *Distributed Transaction Processing: The XA+ Specification* version 2
- [10] IONA Technologies Ltd 1995 *Orbix 2, programming guide* Release 2.0 p 1
- [11] IONA Technologies Ltd 1995 *Orbix 2, reference guide* Release 2.0 p 1
- [12] Sédillot S, Liang J and La Chimia J 1995 Integrating DCE RPC with OSI TP to offer a transactional RPC *Proc. 1st Workshop on High Speed Networks and Open Distributed Platforms (St Petersburg)*
- [13] Houston P 1995 *Transarc Corporation's Distributed Object Strategy Overview (White Paper 11 September)* <http://www.transarc.com>
- [14] Distributed transaction manager Open TP1 <http://www.hitachi.co.jp/Prod/comp/soft1/open-e/opentp1.htm>
- [15] ORBeline 2.0 <http://www.promoco.com/> and <http://www.visigenic.com/prod/orbpd.html>
- [16] Object-oriented transaction processing TPBroker <http://www.hitachi.co.jp/Prod/comp/soft1/open-e/tpbroker/tp-head.htm>
- [17] Group Bull demonstrates the object transaction service for mission—critical applications <http://www-usa.iona.com/PR/Press8.html>
- [18] HP Distributed Smalltalk 4.0 <http://www.rwi.com/smalltalk/products/vendors/hp/>