**电 子 科 技 大 学**

**UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA**

**硕士学位论文**

**MASTER THESIS**

****

**论文题目 CORBA中对象事务服务(OTS)的研究和实现**

**学科专业 计算机应用**

**学 号 3155**

**作者姓名 杨涛**

**指导教师 刘锦德 教授**

摘 要

随着分布式应用规模的日益庞大，分布式系统中不可避免地包括各种异构实体。CORBA提供了分布式环境中对象的透明互操作，使应用程序独立于软硬件平台、网络协议、编程语言，近年来已成为分布式对象计算领域的重要研究方向。对象事务服务(OTS)将事务概念引入到CORBA中，保证了对象互操作的完整性和可靠性。本文从理论和实践两个方面对OTS进行了深入研究。

论文首先研究了CORBA体系结构，包括ORB核心、界面定义语言、界面仓库和实现仓库、语言映射、存根和框架、动态调用、对象适配器以及ORB之间的互操作协议等。其次介绍了事务概念和分布式事务的两阶段提交过程，深入研究了OMG定义的对象事务服务的规范，分析了它的事务处理框架，详细说明了各界面（接口）的地位和功能。

论文随着研究了可移植对象适配器规范，这是实现可伸缩的高性能CORBA服务的关键。在此基础上，采用面向对象的思想进行了系统设计，详细说明了各个类的功能和相互间关系，以及它们是如何协作来实现OTS规范中的事务处理的。

作为早期的分布式事务处理软件框架，X/OPEN DTP模型得到了业界很多厂商的支持，如Oracle, Sybase, Microsoft等。如何使应用程序能够更方便的与支持XA协议的资源管理器进行集成，是OTS实现的一个重要方面。论文最后分析了X/OPEN DTP模型和XA协议，并研究和实现了OTS与XA资源管理器集成的机制。

在以上的研究和开发基础上，为CORBA中间件（NOSEware）具体实现了可实用的对象事务服务（NOSEOTS）。

**关键词：对象；对象事务服务；两阶段提交协议；IDL接口；动态调用。**

Abstract

The coexistence of heterogeneous entities in the large scale distributed systems makes it more and more difficult to develop distributed applications. CORBA provides distributed objects the ability of transparent interaction even if they are on different platforms or written in different language,so recently it becomes an important research field in the distributed computing domain. The Object Transaction Service(OTS) brings the powerful computing concept of transaction processing into CORBA and it assured the integrity and reliability of the interaction between distributed objects. The paper does some research on the CORBA and distributed transaction processing, and an implementation of OTS is presented.

Firstly, the architechure of CORBA is described, including object request broker, interface define language, interface repository and implementation repository, language mapping, stub and skeleton, object adaptor, etc. And the concept of transaction is introduced, including the two-phase commit protocol of distributed transaction. Then it addresses the specification of OTS in detail and the software architechure of transaction processing in OTS is analyzed.

In the following the Portable Object Adapter is studied, which is the key to write scalable and high-performance CORBA server. Using Object-Oriented method it designes the objects in the OTS , describes the functions and relations for each object , analyzes how those objects coordinates to implement the transaction processing.

X/OPEN defined the Distributed Transaction Processing model in 1991 , the XA interface in which was supported by most database manufacturers such as Oracle, IBM, Sybase and Microsoft. Finally how to integrate the XA-compliant resource manager with OTS is studied and implemented.

Based on the research and development given above, an available object transaction service(NOSEOTS) is implemented for a CORBA middleware－NOSEware.

**Keyword: CORBA； Object； Object Transaction； Transaction, Distributed Transaction Processing；**

目 录

[摘 要 1](#_Toc20059)

[Abstract 2](#_Toc26896)

[目 录 3](#_Toc28433)

[第一章 绪论 5](#_Toc9825)

[1.1 研究背景和意义 5](#_Toc362)

[1.3 研究方向与现状 7](#_Toc5782)

[第二章 公共对象请求代理体系结构 8](#_Toc10414)

[2.1 OMA 9](#_Toc28733)

[2.1.1 对象服务 9](#_Toc2037)

[2.1.3 公共设施 9](#_Toc32209)

[2.1.3 域界面 10](#_Toc3236)

[2.1.4 应用界面 10](#_Toc22694)

[2.2.2 IDL语言和语言映射 10](#_Toc28190)

[2.2.3 存根和框架 11](#_Toc22641)

[2.2.4 动态调用 11](#_Toc5045)

[2.3 本章小结 12](#_Toc17379)

[第三章 对象事务服务 13](#_Toc23386)

[3.1 事务基本概念 13](#_Toc29477)

[3.2 OTS事务处理框架 13](#_Toc20130)

[3.3 本章小结 13](#_Toc23359)

[第四章 OTS设计和实现 14](#_Toc279)

[4.1 基于C++的CORBA程序设计 14](#_Toc29105)

[4.2 OTS设计和实现 14](#_Toc27121)

[4.3 OTS事务处理机制 14](#_Toc4541)

[4.4 本章小结 14](#_Toc23715)

[第五章 OTS与X/OPEN模型集成 16](#_Toc6465)

[5.1 X/OPEN DTP模型 16](#_Toc7916)

[5.2 XA接口规范 16](#_Toc24840)

[5.3 NOSEOTS与XA资源管理器的集成 16](#_Toc20553)

[第六章 全文总结和进一步的工作 17](#_Toc5372)

[6.1 全文总结 17](#_Toc13805)

[6.2 进一步的工作 17](#_Toc21161)

[参考文献 18](#_Toc22461)

[致谢 19](#_Toc26845)

[作者攻读硕士期间科研项目、论文发表和获奖情况 20](#_Toc32217)

[一、科研项目 20](#_Toc10620)

[二、完成论文 20](#_Toc23482)

[三、获奖情况 20](#_Toc1726)

[附录 21](#_Toc23787)

[附录A OTS IDL接口 21](#_Toc16229)

第一章 绪论

## 1.1 研究背景和意义

1946年2月15日，美国研制成功世界上第一台通用电子数字计算机ENIAC，揭开了计算机时代的序幕。早期的计算机庞大而又昂贵，大多数机构只有少数几台计算机，人们通过与主机连接的哑终端来使用计算资源，所有的任务都在主机上执行，这时信息资源集中在很少的计算机上。八十年代中期出现了微处理器，并且一直以摩尔定律所揭示的速度发展，目前许多微计算机具有了以前大型机的计算能力，但价格却只是它的几分之一，因此微计算机得到了广泛的普及。随着大量计算机的出现，信息源的数目飞速增长，人们迫切希望具有不受人员、数据以及机器的物理分布限制的资源共享。另一方面，网络技术的发展使计算机互连成为可能，由此产生了将计算任务分解到多台计算机执行的分布式计算模式。



图1-1 OMA的参考模型

随着分布式应用的深化，分布式计算规模越来越大。由于各种各样的因素，大规模的分布式系统中总是存在着各种各样的异构实体，其中包括异构软硬件平台、异构操作系统、不同的编程语言、异种网络协议和异构应用软件等。大规模分布式系统的异构性不仅是不可避免的，也是永远无法消除的。只有异种实体的存在，才能更好地满足不同用户和不同应用的各种不同要求。如果强制性地消除异构实体，也就不可避免地削弱分布式系统的诸多优势。因此，如何使大规模分布式系统中的异构实体能够相互协作，实现跨平台资源的透明互操作和协同计算，一直都是业界研究的热点。过去的研究表明，解决这一问题的关键在于为分布式应用的开发提供好的工具，并为相应的分布式应用的运行提供好的环境。目前支持分布式计算的环境主要有两类：基于过程的分布式计算和面向对象的分布式计算。

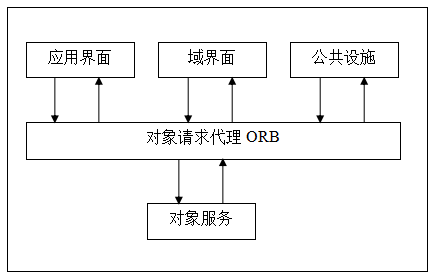


图1.2 CORBA的主要组成部分

基于过程的分布式计算环境主要是OSF协会制订的DCE（Distributed Computing Environment）。OSF是一个非赢利性的协会组织，它的成员包括许多IT和通信公司，著名的有DEC（现已被Compaq兼并）、HP、IBM、Siemens、Sony等。OSF支持分布式计算的思路是，选择并集成所需的软件技术，并向最终用户证明该技术的稳定性和可持续性。



图1-4 事务协调者和参与者逻辑关系图

基于这种思路，OSF采用RFT(Request For Technology)制订了DCE标准。DCE具有传统的客户/服务器体系结构，包括线程、RPC、时间、命名和安全等服务，为分布式应用提供了一个与硬件平台、操作系统和网络无关的开发环境。它刚推出时备受欢迎，曾被工业界视为最好的提供分布式编程环境的软件平台，但随着技术的发展，已经呈现出较多局限和不足，目前业界看好的则是面向对象的分布式计算环境。

对我国软件业而言，CORBA产品的国产化、自主化具有非常重要的意义。首先，它与操作系统、数据库一样，都是重要的系统软件，采用国外厂商提供的系统软件容易带来潜在的安全隐患。其次，国内拥有自主产权的CORBA产品对市场竞争是非常有利的，由于CORBA平台开发难度较大，国外大型软件厂商具有较强的竞争能力，如果完全让国外企业占领国内市场，国内用户只能无条件地使用国外的产品。因此，对CORBA进行研究并开发自己的产品具有非常重要的意义。

表1-1 Current 开始事务接口

|  |  |  |
| --- | --- | --- |
| 返回项 | 类型 | 描述 |
| code | 整数 | 返回值，0代表成功，非零表示失败。 |
| msg | String | 接口调用结果说明 |
| data | String | 接口调用返回的数据 |

## 1.2 研究方向与现状

CORBA规范主要包括三个部分：对象请求代理、对象服务和公共设施。最底层是对象请求代理，规定了分布对象的界面（接口）和语言映射，实现对象间的通讯和互操作，是分布式对象系统中的“软总线”；在对象请求代理之上定义了多项对象服务，如名字服务、事件服务、安全服务等；这些服务是构建很多分布式应用所必须的，并且大多都涉及到关键技术，如安全、事务等。如果分布式计算环境不能提供这些服务，那不仅加大了应用程序开发的难度，有时甚至从技术角度来看是难以接受的；因为很多服务都需要对相关领域有深入的研究才能实现，比如安全、事务处理等。因此，对于基于CORBA的分布式计算平台来讲，提供尽可能多的对象服务的实现，具有十分重要的意义。

表1-2用户增加接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 类型 | 必填 | 描述 | 默认值 |
| 返回项 | 类型 | | 描述 | |
| code | 整数 | | 返回值，0代表成功，非零表示失败。 | |
| msg | String | | 接口调用结果说明 | |
| data | String | | 接口调用返回的数据 | |
| 错误值说明 | | | | |
| Code值 | 描述 | | | |
| 401 | Unauthorized - [\*]：表示用户没有权限（令牌错误或过期） | | | |
| 调用示例 |  | | | |

在公共对象服务中，“对象事务服务”将事务概念引入到CORBA中。事务是包涵一个或多个计算任务的具有ACID属性的操作集合。事务的ACID特性，对构建高可靠性应用，特别是要求并发访问共享数据的分布式应用起着关键作用。事务处理技术在银行、证券、电信等行业已经得到了广泛地应用。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 类型 | 必填 | 描述 | 默认值 |
| 返回项 | 类型 | | 描述 | |
| code | 整数 | | 返回值，0代表成功，非零表示失败。 | |
| msg | String | | 接口调用结果说明 | |
| data | String | | 接口调用返回的数据 | |
| 错误值说明 | | | | |
| Code值 | 描述 | | | |
| 401 | Unauthorized - [\*]：表示用户没有权限（令牌错误或过期） | | | |
| 调用示例 |  | | | |

表1-4 Current开始事务接口

早期的事务处理产品主要是面向集中的大型机，如IBM六十年代推出的支持在线事务处理的CICS(Customer Information Control System)以及基于信息处理的事务处理器IMS(Information Management System)，70年代由Tandem Computers公司推出的事务管理构件Pathway/TS。这些系统都依附于一定的大型机，强调事务处理的完整性，但对用户的交互操作、应用系统的开发和维护方面则没有提供较大的支持。

八十年代网络技术得到很大发展，网络基础设施不断完善，支持各类资源交互和共享的客户/服务器模式以及三层结构的分布式应用得到广泛应用，事务管理器也不再依附于大型机，而以面向开放式系统的中间件方式出现，著名的产品如BEA的TUXEDO和TOP END，IBM的Encina和Microsoft的MTS等。这些产品不仅能够满足事务管理的需要，还提供了负载平衡、系统管理和接近线性加速比的高性能。

第二章 公共对象请求代理体系结构

CORBA是OMG推出的一个重要的工业规范，它是OMA(Object Model Architecture)的核心部分。OMG是一个非赢利性的协会组织，组建于1989年，由八个著名的计算机公司发起。现在它的成员已经超过800个，并且分布很广泛，遍及计算机制造商、软件公司、通信公司和最终用户。OMG的目标是，推动用于开发分布式计算机系统的对象技术的理论和应用。为达到这一目标，OMG所采用的方法是，为面向对象的应用提供一个公共框架，如果符合这一框架，就可以在主要的硬件平台和操作系统上建立一个异质的分布式应用环境。

OMA包括两部分：对象模型和参考模型。对象模型定义如何描述分布式异质环境中的对象；参考模型描述对象之间的交互。OMG要求RFP提交的建议必须符合这两个标准，否则将不予采用。

## 2.1 OMA

### 2.1.1 对象服务

对象服务是与具体的应用领域无关的界面，所有分布式对象程序都可以使用它们。

这些服务极大地丰富了分布式对象的特性和运行支撑环境。

### 2.1.2 公共设施

与对象服务不同的是，公共设施面向最终用户的应用。例如，DDCF（Distributed Document Facility）是OMG 所采用的一个的公共设施，它是一个基于OpenDoc的复合文档公共设施。DDCF允许基于文档模型的对象的表示和交换。

### 2.1.3 域界面

这些界面所完成的任务与对象服务和公共设施类似，但针对某一特殊的应用领域。例如，PDME（）是OMG发出的最早的这类RFP之一，它是为解决制造领域中的问题而发出的。另外，OMG也已经发出了通信、医药和财务等领域中的这类RFP。

### 2.1.4 应用界面

CORBA 是OMG制定的首批重要规范之一。它详细说明了OMA中ORB组件的特性和界面。由公式（2-2）

（2-1）

ORB 的这些特点，使应用开发者不必过多地担心底层的分布式编程问题，从而可以集中精力设计自己的具体应用。

发送请求时，客户利用对象引用来指明目标对象。因为每当创建一个CORBA对象时，同时也就创建了它的对象引用。对象引用让客户能运用它，但不能修改它，因为只有ORB才知道对象引用的内部详情。

（2-3）

## 2.2 CORBA

CORBA 是OMG制定的首批重要规范之一。它详细说明了OMA中ORB组件的特性和界面。

### 2.2.1 IDL语言和语言映射

在客户向目标对象发送请求之前，它必须知道目标对象所能支持的服务。目标对象通过界面定义来说明它所能提供的服务。CORBA的对象界面由OMG IDL语言来定义。

OMG IDL 是一个纯说明性语言，不是编程语言，并且与具体的宿主语言（主机上的编程语言）无关。这就强制性地使界面与对象实现分离，这样就可以用不同的语言来实现对象，而它们之间又可以进行互操作。与语言无关的界面在异质分布式环境中是非常重要的，因为不同的平台上常会支持不同的编程语言。

如上所述，既然不能用OMG IDL 直接去实现分布式应用，那么就需要把IDL描述的特性映射为具体语言的实现，这就是语言映射的任务。到目前为止，OMG已经为C、C++、SmallTalk、Ada 95、COBOL和JAVA的语言映射制定了标准。

### 2.2.2 存根和框架

除了把IDL的特性映射到具体的编程语言外，OMG IDL编译器还根据界面定义来产生客户方的存根和服务方的框架。存根代表客户创建并发出请求；框架则把请求交给CORBA对象实现。具体地说，存根为客户提供了一种机制，使得客户能够不关心ORB的存在，而把请求交给存根，存根则负责对请求参数的封装和发送，以及对返回结果的接收和解封装。框架在请求的接收端提供与存根类似的服务，它将请求参数解封装，识别客户所请求的服务，（向上）调用对象实现，并把执行结果封装，然后返回给客户方。

由于存根和框架都是从用户的界面定义编译而来，所以它们都与具体的界面有关，并且在请求发生前，存根和框架早以分别被直接连接到客户程序和对象实现中去。为此，通过存根和框架的调用被通称为静态调用。

### 2.2.4 动态调用

DSI在服务方的角色与DII在客户方的角色相同。与DII允许客户不通过存根就可以调用请求类似，DSI允许用户在没有静态框架信息的条件下来调用对象实现。

## 2.3 本章小结

本章分析了OMG的CORBA标准，这是一个发展迅速的面向对象的中间件规范。CORBA的关键组成部分包括：ORB核心、OMG界面定义语言、界面仓库和实现仓库、语言映射、存根和框架、动态调用、对象适配器以及ORB之间的互操作协议。另外，在这一部分中，还介绍了与CORBA关系密切的OMA，其中包括OMG的对象模型、参考模型、对象服务、域界面和应用界面。

第三章 对象事务服务

CORBA作为当前分布式计算的重要规范，提供了分布式应用所需的多项服务。在CORBA的公共对象服务中，对象事务服务（Object Transaction Service,以下简称OTS）将事务概念引入到分布式对象计算中。事务是构造可靠的分布式应用的基础，在很多行业如银行、保险、证券等关键性业务中得到了广泛的应用。随着应用的普及和深化，传统的事务处理技术面临新的挑战，如系统规模无限扩大对性能要求更高，业务逻辑的快速变化希望开发周期更短。对象事务服务结合了分布式对象技术和事务管理技术，一方面具有事务管理器的高可靠性，同时又具有CORBA中互操作性、可扩展性、可维护性等优点，提供了良好的分布式事务处理解决方案。

## 3.1 事务基本概念

很多情况下，应用程序都能够正常执行，但在意外掉电等特殊情况下会导致处理失败。对关键性应用来讲，这是不可接受的。一个最典型的例子就是银行的转账业务。如果计算机将一个账户的金额减去一定数量，但还没有将该金额转移到另一个账户之前系统发生崩溃，那就会导致这一部分资金消失，这显然是不允许的。事务则为应用程序提供了保证操作全部可靠完成的机制。

## 3.2 OTS事务处理框架

事务的ACID属性对开发可靠的分布式应用起着重要作用，因此OMG在CORBA公共对象服务中定义了有关对象事务服务的规范，包括服务描述、服务框架、服务应用模式和实现要求。

## 3.3 本章小结

本章首先对事务概念进行了简明的介绍，分析了OTS采用的两阶段提交协议，并指出了它对实现OTS是至关重要的。然后阐明了OTS的体系结构，包括OTS中各个实体之间的逻辑关系，和各接口中操作的语义和功能。最后，阐述了OTS中事务的管理和传播机制，探讨了各个对象是如何协作实现事务处理的。

第四章 OTS设计和实现

OTS作为CORBA公共对象服务，其实现遵循CORBA服务的一般实现原则，包括采用面向对象的思想，通过ORB实现透明的对象互操作，接口和实现相分离等。因此，CORBA框架下的程序开发，特别是服务器对象实现是研究OTS实现的基础，因此，下面首先介绍CORBA程序设计和开发模式。

## 4.1 基于C++的CORBA程序设计

从图2-1中可以看到，CORBA中对象服务是基于对象请求代理之上的，即ORB为对象服务提供了对象之间的透明交互。

## 4.2 OTS设计和实现

OTS通过多个对象互相协作来保证事务的可靠性和完整性。各对象之间的互操作是通过IDL定义的接口进行的。因此，按照CORBA程序开发流程，第一步是定义各对象之间的IDL接口。

要指出的一点是，任何一个实际的系统，都会有大量细致的工作需要做，比如系统设置、监控、管理等。上面我们主要着眼于事务处理的实现，因此，对其它细节工作如如写事务日志的Logger类，利用PortableInterceptor实现系统中事务状态监控和显示的管理器等，它们对一个真正实用的产品来讲是非常重要的，OTS的实现都是很重要的。但对本文来讲，它们不是重点，因此，限于篇幅，就不一一列举了。

## 4.3 OTS事务处理机制

按照CORBA程序设计过程，各对象实现后就可以编写主程序，并与上述各类的实现文件一起编译，链接，生成可执行程序。课题中的OTS实现作为NOSEware的提供事务处理的组件，称为NOSEOTS。

## 4.4 本章小结

本章首先说明了OTS实现所需要的技术基础，包括CORBA程序开发和可移植对象适配器。然后介绍了NOSEOTS所定义的一些IDL接口。第三节中对NOSEOTS中主要对象的功能和接口进行了设计。最后第四节阐述了NOSEOTS中各对象互相协作的事务处理机制，专门列出了事务处理过程中对OTS规范定义的IDL接口的调用，说明NOSEOTS是完全遵循OTS规范的。

第五章 OTS与X/OPEN模型集成

为了使各个不同厂家的产品能够协作实现事务处理，X/OPEN提出了分布式事务处理模型(Distributed Transaction Processing，简称DTP)，该模型得到了很多计算机厂商，特别是数据库厂商的支持。

## 5.1 X/OPEN DTP模型

X/OPEN DTP模型是一个提供分布式事务处理的软件框架，它主要包括下列组成部分：应用程序（AP）、事务管理器（TM）、资源管理器（RM）。

在X/OPEN DTP 模型中，典型的事务处理过程是这样的：应用程序首先通过TX接口告诉事务管理器要开始一个新事务，事务管理器分配全局事务ID，调用XA接口通知各资源管理器新事务开始。然后应用程序访问资源管理器，如执行Update等SQL语句等。当操作完成后，应用程序要求事务管理器提交事务，事务管理器调用XA接口，协调各资源管理器进行提交。

## 5.2 XA接口规范

XA接口规范实际上是以C语言的语法定义的函数接口，支持XA接口的资源管理器称为XA兼容的资源管理器。XA接口主要是一个称为xa\_switch\_t的结构，它定义了RM参与事务处理时需要提供给TM调用的函数(名称全部以xa\_开始)；详细的参考手册可见参考文献6。

## 5.3 NOSEOTS与XA资源管理器的集成

比较OTS和DTP的体系结构可以知道，DTP中的资源管理器与OTS中的Resource地位类似，都表示事务参与者，XA资源管理器要参与OTS协调的事务处理，最根本的就是要将XA接口中的操作映射为OTS中Resource接口中的操作来参与OTS协调的两阶段提交，同时，要保证OTS对XA接口中各函数的调用顺序以及和调用线程之间的关系要遵循XA接口中的定义。

第六章 全文总结和进一步的工作

## 6.1 全文总结

本文从理论和实践两个方面对CORBA中间件所需要的对象事务服务（OTS）进行了较深入的研究和探索。

论文第一章指出，随着CORBA研究的不断深入和事务处理在实际应用中的普遍采用，在面向对象的分布式应用中引入事务处理模型是分布式对象计算和事务处理领域的一个重要研究方向。本章分析了研究和实现对象事务服务的必要性，总结了国内外已有的研究成果，指出了继续研究这一课题的方向。

论文的第五章研究了如何将NOSEOTSOTS与X/OPEN DTP模型中XA资源管理器集成，这对NOSEOTS在实际应用中的推广具有重要意义。本章首先介绍了X/OPEN DTP模型，研究了XA接口规范，对NOSEOTS与XA资源管理器的集成进行了设计和实现。

## 6.2 进一步的工作

对象事务服务结合了分布式对象计算和事务处理，是一个具有相当广度和深度的领域，并且课题自身的特点又决定了，必须从理论和实践两个方面同时对其进行研究。因此，虽然经过近两年全身心的投入，本论文仍然存在着以下几个方面需要进一步的深入和完善：

* 在OTS规范中，事务模型包括扁平(Flat)事务和递归(Nested)事务。OMG规定，OTS实现必须支持扁平事务，对递归事务是可选的。目前NOSEOTS尚不支持递归事务。递归事务可以回滚子事务而不影响全局事务，减少了事务失败的可能性，为应用程序提供递归事务是很必要的。
* 目前OTS服务和应用程序开发包都是用C++语言实现的，这意味着应用程序开发只能使用C++，这是不尽如人意的。由于下层的ORB－NOSEware支持C++和JAVA语言映射，因此完全可以为应用程序提供基于JAVA的开发包。

参考文献

1. Object Management Group. The Common Object Request Broker: Architecture and Specification. 1999.

2. Object Management Group. CORBA services: Common Object Services Specification. 2000.

3. Object Management Group. Object Transaction Service. 2000.

4. Object Management Group. CORBA 2.3 full specification. 1997.

5. Object Management Group. CORBA Facilities Architecture Specification. 1997.

6. X/Open. Distributed Transaction Processing: The XA Specification. 1991.

7. X/Open. Distributed TP: The XA+ Specification, Version 2. 1994.

8. Oracle Corporation. Oracle8i Application Developer's Guide via: <http://technet. oracle.com/>doc/oracle8i\_816/appdev.816, 1999.

9. Sybase Corporation. Using Adaptive Server Distributed Transaction Management Features. October 1999.

10. Sybase Corporation. XA Interface Integration Guide for CICS, Encina, and TUXEDO. October 1999.

11. Microsoft Corporation. MSDN Library Visual Studio 6.0. 1997.

12. Michi Henning, Steve Vinoski 著. 基于C++ CORBA高级编程. 徐金捂，徐科，吕志民等译. 清华大学出版社，2000.

13. 苏森. 面向对象的互操作技术. 电子科技大学1998年博士论文.

14. 邵佩英. 分布式数据库系统及其应用. 科学出版社，2000.

致谢

在本论文即将完成之际，谨向我的导师刘锦德教授致以衷心的感谢！本论文的工作是在刘老师的悉心指导下完成的。刘老师以他敏锐的观察力、渊博的知识、严谨的治学态度、精益求精的工作作风和对科学的献身精神给我留下了深刻的印象，这些使我受益匪浅，并将成为我终身献身科学和献身事业的动力。特别需要提及的是，在我攻读硕士学位期间，导师不仅为我创造了优越的科研和学习环境，使得我的论文工作得以顺利完成，同时在思想上和生活上也给予了我极大的关心和帮助。

在我即将结束十八载校园生涯的时候，我真诚地感谢我的父母，他们用自己的辛勤劳动给了子女受教育的机会，在一个农村家庭培育了三个本科、两个硕士！

感谢我的三个姐姐以及兄长杨渠波对我的关心、鼓励和支持。

衷心地感谢课题组的其他成员——郭乐深博士、曹晓阳博士、曹亚西硕士，他们的密切配合和无私帮助保证了论文工作的顺利进行。

感谢我的同窗好友刘一钫硕士、何洪伟硕士所给予的启迪和诚挚的友谊。

本论文工作的研究得到了国腾道亨研发中心的资助，在此一并深表谢意。

最后，感谢曾经教育和帮助作者的所有老师。衷心地感谢为评阅本论文而付出辛勤劳动的专家和教授们！！！

作者攻读硕士期间科研项目、论文发表和获奖情况

## 一、科研项目

横向项目：国腾道亨公司基于CORBA中间件的开发和应用（项目编号：KW037）············已经完成所承担的分工：对象事务服务的开发。

## 二、完成论文

1. 郭乐深、刘锦德、杨涛，通过有效的端系统动态资源管理实现QoS控制，通信学报，2001.6.

2. 杨涛、郑晓霞、刘锦德，基于CSTA规范的CTI中间件的研究和实现，计算机应用，2001.10.

3.杨涛、郑晓霞、刘锦德，CORBA中对象事务服务研究和实现，电子科技大学学报，2001.12.

## 三、获奖情况

2001年11月获得电子科技大学一等奖学金。

2001年12月获得“三星”专项一等奖学金。

附录

## 附录A OTS IDL接口

#include <CORBa.idl>

module CosTransactions {

// DATATYPES

enum Status {StatusActive,StatusMarkedRollback,StatusPrepared,

StatusCommitted,StatusRolledBack,StatusUnknown,StatusNoTransaction,

StatusPreparing,StatusCommitting,StatusRollingBack};

enum Vote {VoteCommit,VoteRollback,VoteReadOnly};

// Structure definitions

struct otid\_t {

long formatID; /\*format identifier. 0 is OSI TP \*/

long bqual\_length;

sequence <octet> tid; };

struct TransIdentity {

Coordinator coord;

Terminator term;

otid\_t otid;

};