基于分布对象的大规模多数据源互操作机制*)

李 贵[1] 张兆鑫[1] 郑新录[1] 李征宇[1] 张 斌[2]
(沈阳建筑大学信息学院 沈阳 110168)[1] (东北大学信息学院 沈阳 110006)[2]

摘 要 针对大规模多数据源互操作所面临的问题,以分布对象技术为基础,并结合有关标准,提出了一种适合于大规模多数据源互操作的三层体系结构,并采用分层机制解决了互操作系统中不同层次的异构性问题,支持了成员系统的自治性,有效地控制了系统的复杂性。在实现互操作服务的基础上提供了一种柔性的集成环境。

关键词 大规模多数据源互操作,分布对象互操作,对象互操作总线,对象包装

Intreoperation Mechanism for Large-scale Multiple Data Sources Based on Distributed Object Technology

LI Gui[1] ZHANG Zhao-xin[1] ZHENG Xin-lu[1] LI Zheng-yu[1] ZHANG Bin[2]
(Shenyang Architecture University, Shenyan 110168, China)[1] (Northeastern University, Shenyang 110006, China)[2]

Abstract To resolve the interoperation problems in the environment of large-scale multiple data sources, a three-level architecture is put forward. In this architecture the heterogeneous and autonomous problems are resolved in different levels, and the complexity of interoperability system is effectively controlled, and the flexible integration environment is provided.

Keywords Large-scale multiple data sources, Distributed object interoperation, Object interoperation bus, Object wrapping

1 引言

随着信息时代的到来和 Internet 的普及,大规模多数据源之间的互操作是一个日益受到广泛重视的研究课题。实现多数据源之间互操作的复杂性大大取决于系统中数据源的数目和异构程度。在解决复杂性和异构性方面,分布对象技术显示了极大的希望。

分布式对象技术是分布式计算和面向对象技术相结合的产物,是由高速、低价、宽带网来驱动的。全球网络化意味着将有数以百万计的服务器通过洲际网连接起来。但问题是传统的客户/服务器模式不能在有数百万服务器和应用程序的宇宙网上协调工作。分布式对象技术能满足这种新的需求。分布对象技术将应用程序和不同的信息源分割成"智能型"构件,使其在新一代客户/服务器结构上协调运行,并使应用程序能够依据网络结构和应用需求的变化而灵活地组织与重构。

本文以分布对象技术为基础,结合对象管理集团 OMG 的 CORBA 标准,研究了大规模多数据源的互操作机制,提出了一种新的结构、模型和方法,以解决大规模多数据源互操作系统中所面临的各种问题。

2 基于分布对象管理平台的大规模多数据源互操作机制

为实现大规模环境下的多数据源互操作,在深入分析相关研究的基础上,我们提出了一种基于分布对象管理平台的多数据源互操作机制。该机制的基本结构框架如图 1 所示,由 3 个层次构成:第一层为分布对象构件层,每个对象构件代表一个对象化的成员系统,该成员系统采用对象包装技术解决了成员系统的异构性和自治性问题,并通过对象适配器与互操作平台相连;第二层为分布对象管理平台层,为应用系统和各成员系统之间互操作与集成提供了一种基于对象代理机制的面向对象的互操作总线,该平台在解决平台级异构性问题的基础上为所有的成员系统,包括应用系统提供了规范化的统一的互操作接口(包括应用编程接口 API 和接口定义语言 IDL 接口);第三层为互操作服务层,该层将以插件式方式,为满足不同应用领域的互操作和集成需求提供相应的互操作和集成服务对象构件,多数据源互操作系统所需要的基本构件包括模式集成构件、多库事务管理构件、多库查询优化构件等。

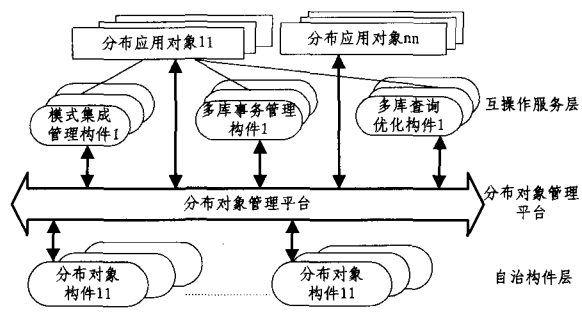

图 1 基于分布对象管理平台的互操作系统体系结构

*)本课题曾得到国家"863 计划"研究项目"基于 CORBA 的多源数据互操作与分布处理系统"资助。**李 贵** 博士,教授,主要研究方向为软件工程、分布对象技术和信息集成技术;**张兆鑫** 研究生,主要研究方向为分布对象技术和信息集成技术;**郑新录** 讲师,主要研究方向为软件工程和信息集成技术;**李征宇** 讲师,主要研究方向为软件工程和信息集成技术;**张 斌** 博士,教授,博士导师,主要研究方向为分布对象技术、XML 数据管理和 Web 服务。

3 分布对象构件

3.1 分布对象构件的结构模型

在大规模多数据源互操作系统中，一个对象构件主要包括两个主要的功能模块：数据源系统、对象包装器，其结构如图 2 所示。

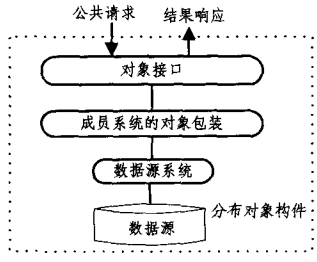

图 2 对象构件系统结构图

数据源系统：在大规模互操作系统中所有要参加集成和互操作的成员系统定义为数据源系统，由数据源、数据管理系统和数据源上已存在的应用程序组成。其中数据源管理系统可能包括各种类型数据库管理系统、文件管理系统、基于知识的专家系统、决策支持系统和各种应用系统等。

数据源系统对象包装：在互操作系统中将不同的数据源系统中要参加集成和互操作的数据或服务以统一的对象模型来描述，对象包装的主要目的是解决成员系统的模型和系统的异构性问题。

分布对象构件：在大规模互操作系统中，已经被对象化的能够提供某种共享服务的数据源系统叫分布对象构件。这种构件具有对象特点，可通过接口访问其服务和信息，它的内部结构和实现对用户来说是隐藏的。

当把一个数据源系统经过对象包装注册到分布对象管理平台上时，对象包装的粒度是非常关键的。如果对象包装的粒度过细，比如关系数据库的一个元组或对象数据库的一组对象，这时就需要互操作系统提供诸如查询、事务处理和并发控制等数据库功能，这既增加了互操作系统的代价，也破坏了成员系统的自治性。如果把整个数据库系统作为对象进行包装和注册，所有数据库的管理和操作功能都保留在对象的内部实现，既减轻了互操作系统的负担，同时也保持了成员系统的自治性。

3.2 成员系统的对象包装

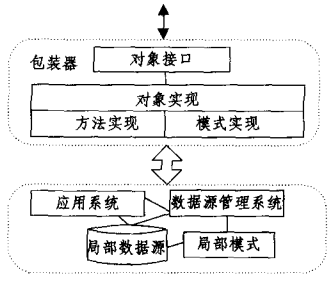

图 3 对象包装器的结构

对象包装所面临的主要问题是在保持数据源系统的自治性基础上解决数据源系统之间的异构性问题，其有效方法是采用面向对象技术将每个数据源系统进行抽象并封装为对象，即采用面向对象数据模型作为公共数据模型，将每个数据源系统所要共享的信息和服务抽象为以公共对象模型表示的对象（数据源对象）。每个数据源系统对象包装的主要成分包括（如图 3 所示）：

数据源对象：以面向对象的公共数据模型表示的数据源系统，即自治构件，它由数据源对象接口和数据源对象实现组成。

数据源对象接口：描述数据源对象所提供的服务，即数据源对象方法的方法签名。数据源对象方法表示的服务是管理和处理数据源中数据的有效方法，它支持诸如持久性、查询、事务等数据管理功能和诸如打印、报表生成等和已存在应用程序的数据处理功能。

数据源对象实现：根据数据源对象接口而编写的基于数据源系统的数据源对象方法的实现代码，包括对象初始化、基于公共数据模型的系统方法在数据源系统上的实现代码、数据源的局部数据模型和公共数据模型之间的映射（模式转换）程序、数据源上的各种应用程序等。

将数据源系统进行对象包装满足了成员系统在互操作过程中的异构透明性和自治性需求：首先，由于被包装后的数据源系统之间的互操作是通过分布对象之间发送消息来实现的，这种发送给数据源系统对象的消息仅取决于其对象的接口，而独立于数据源系统上对象（方法）的实现。这样，数据源系统间平台级和系统级的各种差异被封装在对象的实现中，从而支持了异构透明性；其次，由于数据源系统被独立地和透明地操作和改变，而维持其接口不变，因而满足了自治性要求。

4 分布对象管理平台

分布对象管理平台 DOMP（Distributed Object Management Platform）为实现大规模异构分布环境下资源（信息资源和软件资源）的复用、移植和互操作提供了一种面向对象的互操作环境。DOMP 将一个分布异构系统模型化为一个相互作用的、表示分布系统资源的分布式对象构件的集合，即各种成员系统的资源（信息资源和服务资源）被封装为对象构件；而成员系统提供的服务被抽象为对象构件的方法，方法构成对象构件的接口。一个 DOMP 由任意数量的分布式服务对象（包括系统对象和应用对象）构件和客户对象构件组成，服务对象构件构成了系统的计算资源，客户对象构件通过接口提出由这些资源所完成的操作请求。客户对象构件和服务对象构件之间只是通过接口方式进行交互，具体实现是由底层的对象代理构件和相关的对象服务来完成的，完全独立于客户对象和服务对象构件。对象代理允许客户对象通过接口提出对服务对象的请求，客户对象不需要知道服务对象的位置和实现细节。客户对象和服务对象构件通过相应的软件接口连接到分布式对象管理平台上，通过这些接口把请求和结果转换到成员系统接受的形式。

4.1 基本操作模型与服务模型

4.1.1 基本操作模型

DOMP 的基本操作模型如图 4 所示，主要成分包括：

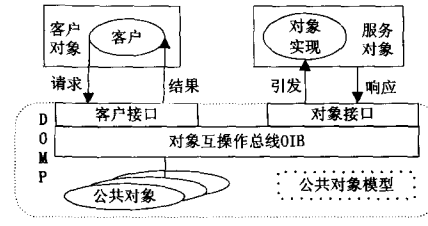

图 4 分布对象管理平台的基本操作模型

(1)客户对象。请求服务对象完成指定操作的实体,可以是应用程序、软件工具,本身也可以是一个服务对象(当其它的对象请求它的服务时)。

(2)客户接口。允许客户请求对象完成的操作、提供操作的参数和接受结果。

(3)对象实现。定义在对象上完成请求服务的计算活动,包括计算请求的结果、更新对象的状态,在处理的过程中可能还会向其它的对象发出服务请求。对象的实现模型包括两部分:执行模型和结构模型。执行模型描述请求服务是怎样完成的,结构模型描述服务是怎样定义的。方法是实现服务的程序代码,方法的执行叫做方法的引发。

(4)对象接口(服务接口)。实现服务对象构件与平台的互连,允许分布对象管理平台引发对象的实现,以完成客户对象请求的功能。

(5)对象互操作。客户对象通过发送消息引发服务对象的操作,实现所需要的服务。在客户对象引发服务对象操作的过程中,应该解决提供相同服务功能的不同对象之间的接口异构性问题,同时要保留对象的语义。

(6)对象互操作总线 OIB(Object Interoperation Bus)。完成服务对象的定位,传递客户请求到指定的对象,并激活对象的实现,完成结果的回送。

(7)公共对象。可通过 DOMP 接口使用的、用于构造各种分布应用和支持平台各种透明性的通用服务,包括基本对象服务和通用设施服务。

(8)公共对象模型。提供从各种异构数据源系统和各种应用系统中构造分布式对象的一种抽象描述机制和这些对象相互作用的机制。

4.1.2 基本服务模型

分布对象管理平台本身是一个分布对象系统,也就是说它提供给客户对象和服务对象之间的互操作功能实际上是由一组相互关联的系统对象所提供的服务来实现的。在整个互操作系统中,所有对象按其所提供的服务类型可分为如下 4 个层次(如图 5 所示)。

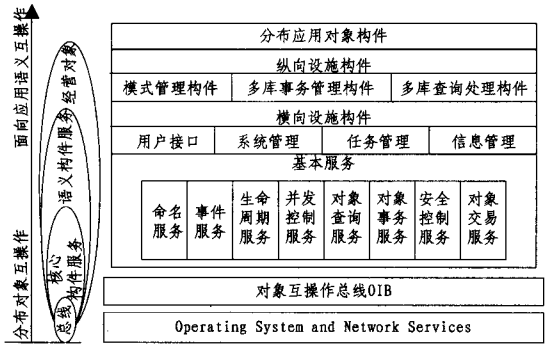

图 5 DOMP 的服务结构

(1)对象互操作总线 OIB:是 DOMP 的核心服务部件,负责系统中所有对象之间的通信服务,使对象能够在分布式环境中透明地请求和响应服务。它是一种逻辑软总线,提供在对象间流动的信息分发、定位和协议数据单元的组装和拆卸。除完成通信服务外,OIB 还要提供最基本的代理服务功能。

(2)基本服务对象层:是 DOMP 服务对象的一个子集,通过 OIB 提供使用和管理 DOMP 中所有对象的基本服务,每个基本服务对象都有明确的接口定义和功能语义,它的实现是由系统本身提供的。基本对象服务功能之间具有正交性,这种正交性允许其它对象同时引发多个基本服务对象所提供的服务功能。这些基本服务对象是以插件方式连接到 DOMP 上的。为有效支持大规模多数据源的互操作,在 DOMP 中提供的基本服务(对象)包括:命名服务;事件服务;对象生命周期服务;并发控制服务;对象查询服务;对象事务服务;交易服务。

(3)通用设施对象层:是 DOMP 服务对象的一个子集,是一组可被多个应用共享的服务对象集合。为在分布对象管理平台上有效开发应用(对象)提供基本的服务构件,这些基本的服务构件分为两大类:横向构件和纵向构件。横向构件是面向所有应用,比如用户接口构件、系统管理构件和任务管理构件等;纵向构件是面向特定应用领域的,比如针对多数据源互操作系统来说,有关模式集成的视图集成构件、全局模式集成构件和联邦集成构件都属于纵向构件范畴里的服务对象。

(4)应用对象层:为用户提供完成指定任务的应用程序的集合,包括客户应用程序和服务器应用程序。应用对象可以使用其底层提供的基本对象服务来构造自己的应用对象。这一层一般不采用标准化方式,以期能在最大自由度上允许开发者使用不同的对象技术,提供不同的对象实现方法。

对 DOMP 基本服务模型来说,OIB 是其核心部分。OIB 与基本服务对象一道确保基于 DOMP 的应用对象之间进行有效的通信与协作。DOMP 的服务结构如图 5 所示。OIB 构成了互操作系统的构件总线,基本对象服务层构成了互操作系统的核心构件服务,通用设施对象层构成了互操作系统的语义构件服务,应用对象层构成了互操作系统的经营对象。分布对象管理平台的发展方向是从支持分布对象互操作到支持面向应用的分布对象协作。

4.2 分布对象互操作的分层结构模型

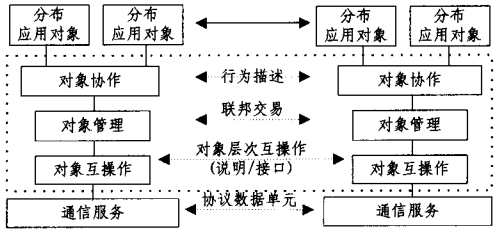

图 6 分布对象互操作的分层结构模型

分布对象管理平台的基本目标是实现客户对象和服务对象的透明互操作。我们认为,大规模分布对象环境下一个有效的分布对象互操作应该包括:支持面向应用语义的对象协作、支持分布对象(服务)的有效管理、支持跨平台的透明对象互操作(包括:不同环境中的客户接口和服务接口的透明匹配及操作捆绑)、支持对象之间消息(包括请求和结果)的透明传送。为此,我们提出了分布对象互操作的分层结构模型(如图 6 所示),为分布对象管理平台的设计与实现提供理论基础。在该结构中,最上层为满足应用需求的分布应用对象层;第二层为对象协作层,该层为构造分布式应用或实现面向任务的对象互操作提供了一种行为抽象机制,即面向应用或面向任务的对象协作机制,这种协作机制是基于规则约束的;第三层为对象管理层,主要功能是实现对象服务的动态管理,包括对象服务的动态发布、动态获取和动态选择,解决大规模环境中的对象服务的定位问题;第四层为对象互操作层,主要任务是解决大规模环境下不同层次的对象互操作,包括说明层互操作、接口层互操作和通信层互操作;最底层为通信服务层,建立在操作系统和网络服务的基础上,并独立于各种操作系统和网络协议,在互操作对象(包括客户对象和服务对象)之间建立通信通道,实现各种协议数据单元的透明传送。

4.3 DOMP 的实现结构模型

DOMP 的实现结构采用一种对象代理机制,主要功能是

为对象之间的通信与协作提供如下透明性机制：服务对象透明定位机制、透明访问路径机制、透明对象数据表达机制、透明消息传送机制和透明的对象操作引发机制。

DOMP 的实现结构分为三大层次：接口层、代理层和通信层（如图 7 所示）。

• 接口层：实现客户对象和服务对象与 DOMP 的互连，主要成分包括：

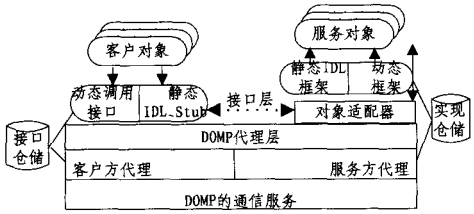

图 7　DOMP 的实现结构

①接口定义语言 IDL。定义接口的操作和相关的属性。使用 IDL 可以定义基本对象使用信息和服务对象的使用信息到接口仓储中。这样，IDL 通过接口仓储将一个特定的服务对象所能提供的操作和如何引发这些操作的信息通知给潜在的一般客户，将一个特定的对象所能提供的服务和如何使用这些服务的信息通知给潜在的应用系统开发者。

②客户接口。客户接口为客户应用提供了动态调用接口和静态 IDL_Stub 接口两种方式调用对象服务。

③接口仓储。接口仓储存储描述服务接口的使用信息，分为两类：基本对象使用信息和服务对象使用信息。

④服务接口。包括对象适配器和基于 IDL 的框架（Skeleton）接口，对象适配器实现了对象实现和 DOMP 的交互，框架实现了 DOMP 对对象实现的调用，它是一种向上的调用接口。

⑤对象适配器。对象适配器既是将数据源对象实现连接到 DOMP 上的一种中件，又是数据源对象存取由 DOMP 提供的服务的一种设施。DOMP 通过对象适配器提供的服务有：(1) 生成和解释对象引用；(2) 方法引发；(3) 注册实现；(4) 映射对象引用到实现；(5) 对象和实现的激活与撤消等。不同的数据源对象要求对象适配器提供不同的服务，因此不能提供唯一的对象适配器或要求所有的对象适配器都提供同样的接口和功能，应该为不同类型的数据源对象实现提供不同的对象适配器，这样可以为特定数据源的对象实现提供对 DOMP 服务更有效的存取。数据源对象依据所要求的服务类型来选择使用哪个对象适配器。在一个互操作系统中一般应提供多种类型的对象适配器，如文件系统对象适配器、关系数据库系统对象适配器、面向对象数据库系统对象适配器、信息检索系统对象适配器等。

⑥实现仓储。存储有关 DOMP 定位和激活对象实现所需的信息，以及有关对象实现的安装和对象激活规则的信息，同时实现仓储也是储存调试信息、管理控制、资源分配、安全性等同对象实现相关的信息的公共区域。

• 代理层：DOMP 代理层的主要功能是定位服务对象，启动并激活它准备接受客户的服务请求，然后将客户的操作请求传送给它，运行完毕将结果回送客户。

DOMP 的代理层也分为两大部分：客户方代理和服务方代理。

客户方代理提供的主要功能是：接收客户对象请求；依据接口仓储实现目标对象服务的定位，然后发送请求并接受从服务方代理返回的结果。在发送请求和接受返回结果时要进行协议数据单元的组装和拆卸；

服务方代理负责对象实现的管理，负责建立和解释对象引用，将对象引用映射到对象实现。当一个新的对象实现创建后，它要向 DOMP 注册，负责在它未运行时启动它；若对象实现的方法或状态不可存取时，服务方代理首先要激活对象实现，即将实现的方法或状态拷贝至运行上下文，服务完成后还须将其撤消。服务方代理接受到客户请求后，依据请求中的对象引用激活对象的实现，完成指定的操作，并将执行的结果返回客户方代理。在接受请求和发送结果时也要进行协议数据单元的拆卸和组装。

• 通信层：DOMP 通信层建立在操作系统和网络服务的基础上，并独立于各种操作系统和网络协议，在互操作对象（包括客户对象和服务对象）之间建立通信通道，实现各种协议数据单元的透明传送。

5　互操作服务层

对于基于分布对象管理平台的大规模多数据源环境下的模式集成问题需要考虑的两个关键因素是：大规模问题和基于分布对象构件问题。针对这两个关键因素利用传统的全局模式和联邦模式集成方式是行不通的。因为在大规模环境下，建立一个全局模式或维护一个全局联邦字典是非常困难的。在大规模环境下不同的企业、部门或组织，以及不同的分布应用系统对不同范围和不同领域的数据源有不同的集成与互操作需求，这决定了在大规模多数据源环境下的模式集成必须采用一种柔性的集成机制，即将数据源从不同的角度，按不同的应用需求划分为不同的领域，然后根据每个领域的具体需求实施不同的模式集成机制。模式集成机制包括：全局模式集成、联邦模式集成和视图集成等。分布对象管理平台的插件式应用模式为开发不同的集成服务对象构件（包括模式集成构件、事务处理构件和查询处理构件等）提供了可能。互操作层所面临的是以公共对象模型描述的不同的分布对象构件，所以互操作层的模式集成必须采用面向对象的机制。由于面向对象的抽象与封装的特点，底层数据源系统呈现给其它成员系统和分布应用系统的只是一个抽象的面向对象的逻辑视图，全局互操作层的用户和分布应用系统只能通过对象的抽象接口访问底层分布对象所提供的服务。所以，互操作层的集成在一定程度上体现了对象接口集成，采用面向对象的视图集成是解决面向对象模式集成的一种很有效的方法。

结束语　本文根据大规模多数据源互操作系统的特点和所面临的问题，以分布对象技术为基础，并结合有关标准，提出了一种适合于大规模多数据源互操作的三层体系结构，采用分层机制解决了互操作系统中的不同层次的异构性问题，有效地控制了系统的复杂性。在实现互操作服务的基础上，提供了一种灵活的集成环境。

参 考 文 献

[1] Milliner S, Bouyuettaya A, Papazoglon M. A Scalable Arthitecture for Autonomous Heterogeneous Database Interactions // Proceedings if the 21st VLDB Conference. 1995

[2] Bouguettaya A. Large Multidatabases : Beyond Federation and Global Schema Integration // Proceedings of the 5th Australasian Database Conference (ADC' 94). Christhurch, New Zealand, January 1994

[3] Dogac A, Dengi C, Kilic E. A Multidatabase System Implementation on CORBA // Proceedings of 6th Inter. Conf. on Research Issue in Data Engineering. 1996

[4] Murphy J, Girmson J. Multidatabase Interoperability in the Ju-

piter System. Information and Software Technology, 1995: 503-513

[5] Bouguettaya A, Papazpglou M. On Building A Hyperdistributed Database. Information System, 1995, 20(7): 557-577

[6] Canal C, Fuentes L, Pimentel E, et al. Extending CORBA Interfaces with Protocols. The Computer Journal, 2001, 44(5): 448-462

[7] Troya J M, Vallecillo A. Controllers: ReusableWrappers to Adapt Software Components. Information and Software Technology, 2001, 43(3): 189-202

[8] Canal C, Fuentes L, Pimentel E, et al. Adding Roles to CORBA Objects. IEEE Transactions on Software Engineering, 2003, 29(3): 242-260

[9] Moreno N, Vallecillo A. A Model-based Approach for Integrating Third Party Systems with Web Applications// Proc. of the International Conference on Web Engineering (ICWE 2005). LNCS 3579. Sydney, Australia, Springer-Verlag, July 2005: 441-452

(上接第 261 页)

通过计算出来的知识文档重要性结果，然后对所有文档按降序排列，就可以将最有价值的知识文档排在前列。

4 系统实现

限于篇幅，整个系统各个模块的设计以及实现，这里不做详细介绍了。下面以简单图示的方式给出本系统的核心模块——知识地图模块的类设计和实现界面。知识地图模块分为两个部分：专家网络和搭建测试项目团队。普通用户在该模块可以编辑自己的项目经历，在编辑时可以选择项目导入数据，如果项目不存在，则可以自己编辑项目，然后导入项目数据，再编辑自己的项目经历，包括使用的技术、工作时间，项目职位、项目规模等等。知识分析员有权限选择其他用户编辑项目经历。在项目经历编辑完成之后，系统会根据用户对某些技术的使用时间来自行定义用户在这些知识点的知识程度，但是用户的知识程度最多达到熟练级别，要想达到精通和专家级，必须由知识分析员来编辑用户的知识程度。图 5 是系统中专家网络定义模块的编辑用户知识程度活动图。

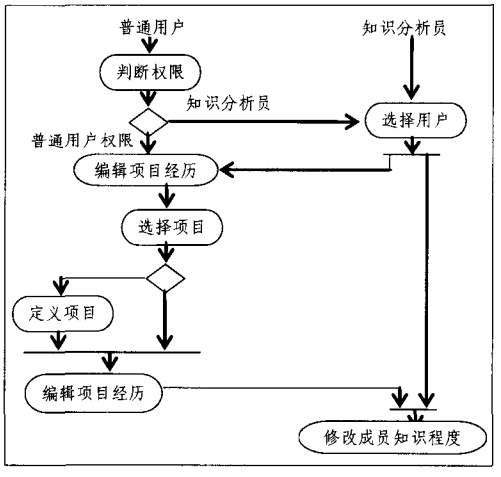

图 5 编辑用户知识程度活动图

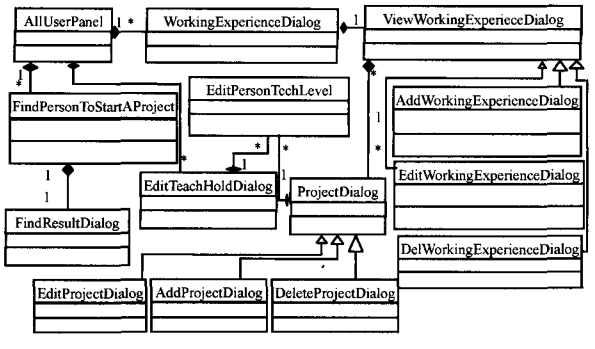

图 6 知识地图模块类图

知识地图模块的客户端的设计类图如图 6 所示。在知识地图模块的客户端中，负责界面消息事务处理的类有三个：EditTechDialog，WorkingExperienceDialog，FindPersonToStartProject。EditTechDialog 这个类用来编辑用户的知识程度，只有知识分析员才有权限来调用这个类。WorkingExperienceDialog 是普通用户可以使用的一个类，用来显示用户的工作经历。类 FindPersonToStartProject 是管理人员，通过输入一些数据来得到组织中比较适合新项目的人员，通过类 FindResultDialog 来显示查找结果。

结束语 知识管理的出现为我们提供了一种新思路和解决问题的新方法，但是软件测试有其自身的特点。虽然现有的通用知识管理理论及技术已或多或少触及了某些问题，但我们更需要用一种与本领域结合更紧密的理论和技术，来重新思考和审视我们的问题，以便寻找出一种解决问题的更有效的方法[9]。进行软件测试领域知识管理的研究，对于提升软件测试组织的整体测试水平和软件企业的整体应变能力，从而最终提高软件产品的质量和企业的经济效益，加强软件企业的核心竞争力无疑具有重大意义。

本文在分析目前软件测试领域存在的知识管理问题的基础上，实现了一个面向软件测试过程知识管理系统，对软件测试领域实施知识管理具有一定的意义，对其他领域的知识管理也有一定的参考价值。它是一个面向软件测试过程的知识管理雏形，虽然系统中还有待进一步完善，缺少对邮件系统和消息系统的支持，以及图形化的统计工具的支持，但是已经具有了一定的使用性，并在实际项目 QESuite2.0 中得到了检验，可望在不久的将来进行商业化和产业化。

参 考 文 献

[1] 夏敬华. 不同视角看知识管理技术[M]. AMT, 2002. Mayer G J. The art of software testing[M]. John Wiley, New York, 1979

[2] Johnson A. An introduction to knowledge management as a framework for competitive intelligence [M]. International Knowledge Management Executive Summit, California, 1998

[3] Rus I, Lindvall M, Sinha S S. Knowledge Management in Software Engineering A State-of-the-Art-Report[M]. The University of Maryland, 2001

[4] 何智涛. 面向软件测试过程的知识管理系统的研究与实现(D). 北京：北京航空航天大学，2003

[5] Gallupe R B. Knowledge Management Systems: Surveying the Landscape[M]. Queen's University at Kingston, October 2000

[6] Microsoft. 数字神经系统——实践知识管理 Microsoft 白皮书[Z]. 北京：微软(中国)有限公司，1999

[7] Tulachan P V. EJB 组件开发指南[M]. 肖国尊，马擎予，译. 北京：清华大学出版社，2002

[8] Software Engineering Body of Knowledge 2004 Version[M]. IEEE

[9] Lawton G. Knowledge Management: Ready for Prime Time? [J]. IEEE Computer, 2001, 34(2): 12-14