计算机科学2001Vol 28№.2

基于对象适配器的协作式信息共享

Cooperative Information Sharing Based on Object Adapter

孙海燕　王晓东　邹　鹏　周　立

（国防科技大学并行与分布处理国家重点实验室　长沙410073）

Abstract　Information sharing is a dominating problem in Computer Supported Cooperative Work (CSCW). In this paper, we discussed the limitation of traditional sharing mechanism in cooperation and proposed an information sharing mechanism based on object adapter in object model. In this mechanism, supports for access control, concurrent control and group awareness can be provided by some object adapters.

Keywords　Cooperation, Information sharing, Shared object, Object adapter

1　引言

计算机支持协同工作(CSCW)是在分布式计算环境下支持群体协同完成一项共同任务的技术，其中全局信息在协作成员之间的共享以及交换是协作的前提。信息共享一直是分布式应用中着重解决的问题，数据库就是实现多用户对共享信息的查询与管理的一种有效途径，然而传统的信息共享还不能有效地支持CSCW系统中的协作，还需要在共享的基础上扩充相应的协作机制。本文以面向对象的建模分析方法为基础，结合协作应用的需求，就基于对象适配器的协作式信息共享进行了研究，文中介绍了传统的对象模型及其在各种协作系统中的应用情况，对协作应用进行了分析，总结了它对信息模型的特殊要求，讨论了利用对象的多种适配器在协作应用中支持协作的信息共享机制，最后指出了下一步工作的方向。

2　对象模型及其应用

2.1　对象模型

面向对象的建模分析方法被广泛应用在许多计算机系统中，并建立了许多各具特色的应用模型，对象是这些模型当中的核心概念：

·应用模型由具有状态、行为以及唯一标识的对象构成；

·对象的状态由它在一组属性上的取值来定义，对象的行为由该对象可执行的操作描述；

·对象通过它所具有的功能接口向其它对象提供服务；

·对象通过类来划分，属于同一类的所有对象具有相同的行为和功能，并在共同的状态域中取值；

·对象的类之间具有子类与超类的关系，超类中定义的行为和功能为子类中的对象所继承。

2.2　协作系统中基于对象的信息表示

面向对象思想的成熟，尤其是面向对象数据库的发展，使对象模型被广泛用于信息的表示。信息是CSCW协作应用中协作行为的载体。协作系统根据不同的时空特性可以作出如图1所示的划分。

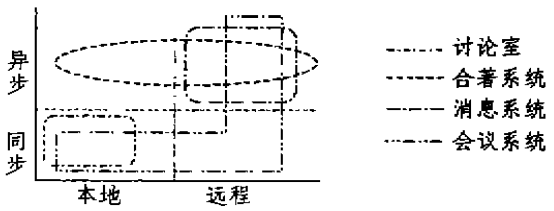

图1　CSCW应用的分类

消息系统是早期用户在远程异步方式下的协作方式。协作成员通过消息通信交换协作信息，系统中用对象来刻画文档、消息等通信实体，并进一步构造邮箱等复合对象。系统中文档是一个具有唯一标识的文本对象，包含用户记录的文本信息；消息是一个被装入"信封"的文档，"信封"中包括作者、收件人、发送时间等属性；邮箱是一个消息对象的有序集合，包含了一个用户所收到的消息按时间顺序排成的序列。典型的消息系统有KOMEX、COSMOS等[1]。

会议系统为多用户就某项会议主题的交流讨论提

孙海燕　硕士研究生，主要研究方向为分布计算技术。王晓东　博士研究生，主要研究方向为计算机支持协同工作。邹　鹏　教授，博士导师，主要研究方向为高级操作系统、分布式计算等。周　立　副研究员，主要研究方向为分布计算技术等。

供协作平台。会议系统可以看成是消息系统的进一步扩展，即原来发给某个收件人的消息被转发给有关该消息的会议。会议可以用一个复合对象来描述，其中包括会议名称、与会者对象集合以及有关的消息对象集合，每一个与会者都具有各自的邮箱，即一个他所收发消息的有序集合。系统中会议的召开与结束可以用会议对象的创建和删除来实现，会议中与会者可以动态地加入或退出，每个与会者通过消息所作的"发言"可被同时发送到所有与之相关的会议对象，并被分发给这些会议的所有与会者。会议系统就是通过管理会议对象为多用户建立协作式的信息交流平台。典型的会议系统有 COM/PortaCOM[2]。

合著系统为多用户提供了一个合同编辑的平台。系统采用对象模型刻画被合同编辑的文本，用户通过对文本对象的复制和加工完成各自的编辑，系统通过对分布对象的并发控制和一致性维护为多用户间的协作建立共享的一致化全局。此外，用户之间还可以通过显式的信息通信传递信息。典型的合著系统有 Quilt[3]。

采用对象模型表示协作应用中的信息，具有以下一些特点：①协作系统中不同形式的共享信息可以用不同类型的对象表示；②信息对象之间存在相关性，是表示协作成员之间协作关系的信息基础；③协作系统中允许对象的动态产生和管理。

上述协作系统中的对象模型虽然可以刻画协作应用中的信息，但没有为应用提供有效的协作支持。我们需要考虑的是建立一个能支持协作的对象模型。

3 协作应用中的信息共享

为了支持应用中信息的协作式共享，信息模型还需要考虑这样几方面的问题：①应用中的信息必须同时满足多用户的动态访问；②应用中对信息的并发控制不会受到传统模型中对共享对象的透明性限制；③建立协作机制，使一个用户的操作能为其他共享用户所感知。

3.1 动态访问控制

多用户信息系统大多使用基于角色的访问控制机制来限制不同用户对共享信息的访问，其一般形式为访问矩阵，即通过一个矩阵说明若干三元组(Subject，Object，Right)中规定的一个用户 Subject 对信息对象 Object 具有 Right 类型的访问权限，角色正是根据用户所具有的访问权限以及对系统所作的贡献作出分类。用户所拥有的权限通常在该用户加入系统时即由系统管理员分配，并且不会在操作过程中频繁变更。因此这种基于角色的访问控制属于静态的控制机制。

在协作应用中，协作者对信息的访问权限会因需要而动态变化，例如会议系统中的一位旁听者变为一个发言人；协作者在整个系统的不同子问题中因为不同的角色而具有不同的访问权限，例如合著系统中担负本章编辑工作的编辑同时可以完成其他章节的校对工作。因此，协作应用中的访问控制应该做到：

·协作者的访问权限不仅可以由系统在初始化中设定，而且可以根据系统状态的变化或者协作者角色的改变动态变更；

·访问控制的粒度应比访问矩阵的控制粒度更细；

·共享对象在具有不同访问权限的协作者之间具有不同的表现形式，这种形式上的不同由协作者的访问权限所决定。

3.2 并发控制

分布式应用中广泛存在着多用户并发访问同一共享对象产生的系统一致性问题，并发控制是解决这一问题的主要途径。

传统的并发控制机制有分布系统中的"锁"(Lock)机制和数据库中的事务(Transaction)机制。锁控制机制是通过对共享数据"上锁"的办法，强制性地将存在冲突的多用户并发访问串行化成顺序的访问序列；事务机制是通过控制数据库中具有原子性、隔离性、持久性、可串行性、可恢复性的事务，向所有用户提供一致的全局数据视图。这些机制的共同之处在于"隔离"不同用户，避免其间冲突的操作可能造成的对系统一致性的破坏。然而，协作者之间的互相感知是协作的前提，所以协作应用中需要具有感知能力的并发控制。为此当前主要有两个研究方向：一是系统中同时建立不同层次、不同粒度的锁，满足不同用户间复杂控制的需要；二是扩充数据库中的事务模型，以满足协作的需要。

3.3 协作机制

协作应用与一般多用户系统最大的区别就在于：传统的多用户系统利用多用户之间的透明性实现信息的共享，用户之间"感觉"不到对方的存在；而协作应用中的每个协作者都必须根据系统中其他协作者的状态彼此协商以完成协作，各成员之间的感知(awareness)是协作的基础。

在同步式协作应用中，成员之间的协作可以借助对共享信息对象的并发访问控制来实现。并发控制在控制多用户操作的同时提供有关用户的具体信息，从而使发生冲突的用户能够感觉到对方的存在，为其协作建立了基础。

在异步式协作应用中，成员之间通过具有历史信息的共享语境提供相互之间协作所必需的信息。为了将时间信息融合到系统中，需要在信息模型中建立时间机制，如数据库中的时态事务模型，以支持异步协作。

4 基于适配器的协作式共享机制

对象通过接口向外界提供功能服务，而对象在多

用户之间的共享控制和管理则可以通过一类特殊对象——对象适配器来实现[1,5]。对象适配器在用户及其所使用的共享对象之间建立间接的关联。它在接收用户提出的对象服务请求后，根据对象的当前状态、用户所需的对象服务类型以及用户所处的应用语境，确定向用户提供的服务接口（如图2所示）。对象适配器一般表现为一组映射规则，描述基础对象具有的功能接口与适配器的功能接口之间的映射。这种映射可以通过以下几种机制来建立：

·过滤：根据用户所处的语境以及对象的当前状态，在适配器中有选择性地建立适配器功能接口与共享对象功能接口之间的映射，共享对象中无效的功能接口则被忽略；

·语义扩展：在适配器中对共享对象的功能接口加以相应修改，以扩充其功能语义，使其旧有功能满足协作应用的需要；

·接口扩展：在适配器中增加新的功能接口，方便用户对共享对象的管理和控制。

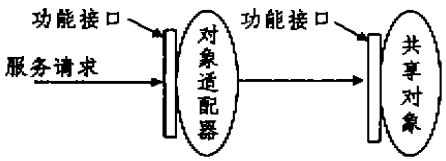

图2　对象适配器示意图

4.1　表示适配器

对于具有不同操作权限的用户，共享对象应该具有不同的功能语义。为共享对象设置表示适配器，可以灵活地为不同用户提供所需要的功能服务。

表示适配器中存储不同用户对共享对象的访问规则，建立用户角色、系统当前状态与该用户对共享对象所允许的操作权限之间的映射。用户在使用共享对象时，表示适配器根据用户当前工作的语境以及适配器中的访问规则判断这一操作是否可以执行，对于允许执行的操作，则向用户提供相应功能接口，否则予以拒绝。这种由用户角色以及工作语境决定其操作权限的机制可以方便地实现对共享对象的动态访问控制，用户角色的修改以及工作语境的变化都能够导致用户拥有对共享对象的不同的操作权限，无需系统作相应的进一步说明；此外，提供表示适配器对共享对象进行访问控制，具有比访问矩阵粒度更细的控制能力。

4.2　锁适配器

基于锁的并发控制机制可以有效地解决共享对象的一致性问题，但由于阻碍了协作成员之间的感知，而且锁控制策略需要由系统静态设置，不能适应协作应用动态、灵活的需要。将锁控制机制应用于对象适配器，可以满足协作应用对并发控制的要求。

系统为共享对象设置锁适配器，保存各种并发控制策略以及当前锁的状态。任何用户对共享对象的访问，都要经过锁适配器根据用户角色、当前锁的状态以及并发控制策略来决定是否给予响应。将并发控制策略存储于锁适配器中，可以灵活地设置控制策略，实现控制策略与控制机构的分离，简化共享对象的控制语义。为了向用户提供必要的感知信息，在锁适配器中还可以保存当前使用该共享对象的用户信息，在并发控制基础上为用户提供共享对象的具体使用情况，在协作成员之间建立相互感知的渠道。

4.3　事件适配器

在协作应用中，各协作者的操作互相影响，紧密联系。协作者需要根据系统的全局状态来完成自己的操作。传统对象模型中对象状态的变化要通过调用其功能接口才能为其他对象所感知，协作成员必须轮询所有相关对象的状态才能确定自己的操作，开销巨大，效率极低，不能满足协作应用的需要。

协作成员之间的感知可以利用事件适配器来实现。事件适配器为其所适配的共享对象产生一条"事件"消息来记录用户的操作引起的共享对象状态的变化，并将其发送到系统中专门的事件处理器。事件处理器中保存了系统预先登记的用户相关性集合，具有协作关系的用户形成一组。在接收到某个对象发出的"事件"消息后，事件处理器根据相关性集合将该事件传递给同组中所有相关用户，通知这些用户系统状态的变化，辅助其确定各自的协作策略。

结论　协作应用在并发控制、群体感知等方面对共享对象模型提出了新的要求。对象适配器在实现对象功能接口的灵活管理基础上，以不同功能的适配器为协作应用提供了有效支持。由于对象适配器在用户与共享对象之间增加了一个管理层，在系统中引入了一定的时间开销。如何在实时性要求较高的协作应用中有效地利用对象适配器将是我们的下一步工作。

参考文献

1　Araujo R B, et al. The Architecture of the Prototype COSMOS Messaging System. In: Proc. of Euteco'88, Elsevier Science Publishers B. V., North-Holland, Vienna, Austria, 1988. 17～170

2　Palme J. COM/PortaCOM Conferencing System: Design Goals and Principles. In: Proc. of IFIP INTERACT'84: Human-Computer Interaction. 931～932

3　Fish R S, et al. A collaborative tool for cooperative writing. In: COIS'88 Proc. of Conf. on Office Information Systems. California, 1988. 23～25

4　The Object Management Group (1990) Object Management Architecture Guide. OMG Document Number 1990. 5. 4. Available through the OMG

5　CORBA The Common Object Request Broker: Architecture and Specification (1991) OMG Document Number 1991. 12. 1. Available thorugh the OMG

6　Mariani J, Roddon T. Cooperative Information Sharing: Developing a Shared Object Service. The Computer Journal, 1996, 39(6): 455～470