计算机科学2000Vol. 27№. 3

一个协同开发框架中的用户协同层模型*)

The Model of the User Cooperation Layer in LCSDM

黄丽雯 李成锴 周笑波 谢 立

(南京大学计算机软件新技术国家重点实验室 计算机科学与技术系 南京210093)

Abstract Two problems in developing a CSCW system are low abstraction level of the cooperation specification and the specification embeded in the program computing part. We design and implement a high level role-based cooperation specification tool UCL. It can be used to discribe the user's relation in a CSCW system to facilitate the developing of cooperative applications.

Keywords CSCW, Cooperation, Distributed system

1 引言

计算机支持协同工作CSCW(Computer-Supported Cooperative Work)以辅助人与人在日常工作中的交互为目的,支持一组用户参与一个共同的任务,并提供给他们访问共享环境的接口,由此为这个用户群提供协同支持。它反映了人们对计算机功能需求的改变,即希望计算机从传统的解决计算问题发展为辅助用户的交互活动。对于计算机支持协同工作要求尽可能体现用户之间的协同以方便用户的协同工作。所以对协同关系的描述的研究一直是协同工作的一个热点。以前的研究着重于通过设计一种专门的协同描述语言并实现其编译系统来进行用户协同关系描述,用户可以用这种专门语言来编写协同程序,如DCWPL,CSDL[1-2]。

CSDL:是一个为协同系统而专门设计的语言,提出了一个建立协同应用的完整环境,它支持动态和静态两种模式在已有的单用户程序的基础上建立协同系统,是一个支持协同特征定义和协同系统的逻辑结构的定义的描述和设计语言。但是它提供的描述方式比较低级,不够抽象,只提供了有限的对用户层次协调定义的支持,迫使程序员在低层次上建立此种定义。

DCWPL:针对以往的研究对用户描述和协同机制的支持较少而研制的协同系统工具,它也是一个专门的解释性语言,它把协同程序的协同部分和计算部分分开,方便了程序员编写协同程序,但是由于协同部分和计算部分用不同的语言实现,程序的接口就比较复杂。因为是专门的语言,所以可移植性较差,而且要求程序员学习新的语言,不方便使用。

总的来说以往的研究存在以下一些缺陷:

- 依赖于具体的应用[5],很多协同描述直接嵌入在应用系统的计算部分中,不能脱离具体的应用而存在,没有通用性。
- 缺乏高层(user-level)的协同描述方法[2],很多描述从比较低级的层次对协同进行描述,虽然它们的描述比较完善,但由于抽象的层次不高,给用户的使用带来了很大的不便。

文中对这些问题进行了探讨,并在此基础上提出了用户协同层UCL模型对用户之间的协同关系进行描述,以提供更高层次的协同描述方法。

2 系统模型概述

考虑到现有的协同系统存在开发和维护困难,不易移植和复用以及协同效率较低等问题,我们提出了一个层次化基于对象的CSCW系统结构模型LCSDM[3a],其结构如图1所示,各模块的功能说明如下:

· 对象协同层 是整个开发系统的基础,这个层次的主要功能是在具有协同关系的对象之间建立联系,实现自动协同,向程序员屏蔽协同的过程及其细节,从而使程序员可以致力于其他处理。它包括对由协同对象构成的协同系统框架的支持和完成对象协同功能的协同代理。

*)得到国家"863"高科技计划(863-306-02-07)基金资助。

黄丽雯 硕士生,主要研究方向为分布式计算、CSCW,李成锴 硕士生,主要研究方向为分布式计算、CSCW,周笑波 博士生,主要研究方向为分布式计算,谢 立 教授,博士生导师,副校长,主要研究方向为分布/并行系统。

· 用户协同层　用户协同以对象协同为基础，构造协同系统中多用户之间的协同关系。协同表达了用户行为之间的依赖关系。在协同系统中，一个用户动作的协同结果取决于该用户在协同系统中所处的地位，即他的协同角色，具体地说，就是由其在各协同表达特性中所处的子角色决定。

· 应用协同层　应用协同以用户协同为基础，用用户协同所描述的各种角色的一个或多个实例构造，并通过具体的应用规则，构造协同系统中多用户之间的协同关系。应用协同的定义包括，用户构成：描述协同应用由哪些用户构成。应用规则：描述协同应用的具体协同规则。会商管理：描述对用户的加入、退出等的管理。

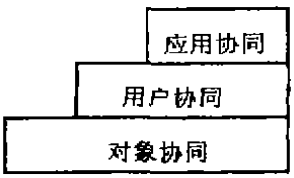

图1　LCSDM 的结构

3　用户协同层 UCL

由于对象协同层的抽象程度不高，对象的分布比较分散，而且对协同关系没有提供描述手段，因此我们在此基础上更进一步抽象提出用户协同层的模型，并为用户提供协同关系描述。考虑到参与协同工作的用户在工作中担任着各种角色，他们之间的关系是多种多样的，这些直接影响着他们之间的协同交互。因此，我们把组和角色作为用户协同层的设计基础。

3.1　组(Group)

人与人之间的关系会随着环境的变化而改变，如：在家里是母女关系的两人到了学校就有可能变成教师和学生之间的关系，这并不是否认了她们的母女关系，而是在不同的环境中占主导地位的关系是不同的。所以要确定描述一个协同系统中的协同关系不能脱离协同系统的大环境。

为了对协同环境进行描述，我们引入了组的概念。在日常生活中，一同为某个目标工作的人的集合就构成了一个组。在协同系统中组即为参与一次协同工作的成员的集合。而在一个组里，所有的用户又是按一定的关系组织在一起的。经过分析，我们发现虽然各个具体应用互不相同，但是用户之间的关系大都可以抽象为有限的几类。我们按在系统中用户的地位和作用的关系将它们归为以下三类：

● 对等的协同用户关系　不管用户的具体权限如何变化，每个用户的权限均相同。用户之间的关系是对等的，如图2。用户之间只有加入的时间先后关系，所有的用户都是合作者。典型的对等协同应用程序如网络下棋，网络聊天，白板。

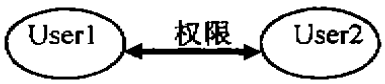

图2　对等关系

● 主从的协同用户关系　系统中的用户是不平等的(图3)，具有不同级别的权限。在一个系统中有一个用户具有最高的地位，称为管理者。他可以把其他用户加入工作组，也可以让其他用户退出工作组。他也可以把管理者的角色转交给其他用户。系统中的一般用户称为协同者，他们在工作组中协同工作，具有平等的地位。在系统中也可以有一部分用户比其他用户的地位低，他们称为观察者(watcher)，他们并不直接参与协同工作，只是可以观察到工作的进程和结果。典型的主从协同应用程序包括协同文本编辑、协同绘画、远程会议等。

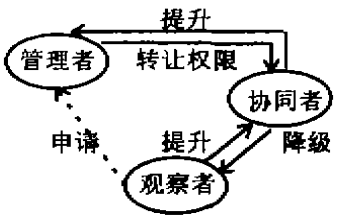

图3　主从关系

● 客户/服务器模式的协同用户关系　系统中的用户权限是不相同的，但不同用户之间没有明显的主从关系(图4)。一部分用户为其他用户提供某种服务，称为服务者。一部分用户接受服务者所提供的服务，称为客户。典型的客户/服务器协同应用程序如虚拟实验室、虚拟教室等。

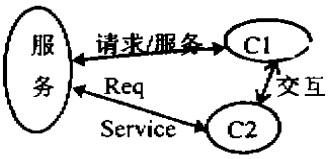

图4　Client/Server 关系

对于那些复杂的协同应用系统的用户关系，基本上可以由以上三种协同模式嵌套组合而来。此外，协同

策略中的等待策略和并发控制也是由一组用户之间的相互关系决定的。因此有必要在组中对它们进行描述。组的具体定义应该包括3个方面：①基本属性：组名、组的大小(最大允许用户个数)、成员的名，成员的类型。②协同模式：对等、主从或客户/服务器模式。③协同策略：它包括等待策略和并发控制。前者描述组内对发生数据访问冲突时的解决策略，如 FIFO，LIFO，High Priority First 等，并发控制描述对给定对象采用哪一类并发控制方法，它又包括：冲突避免法：执行前检测，避免发生冲突，如锁机制，Floor 控制等。冲突检测法：先执行用户动作，然后检测是否发生冲突。如操作转换等。

3.2 角色(Role)

在组的定义中，我们只是提供一个用户之间的协同关系的抽象描述，而在具体的系统中还需要更进一步定义一些具体的权限。这正是用户角色的体现，如：管理员、作者、监督者等等其主要区别就在于个人的权限不同。我们把用户的角色抽象为赋予一组用户的特权的集合。通过对这些用户特权的定义，我们提供了描述用户的权限的手段。这些特权可以包括对某些数据的存取控制和对某些行为(方法)的执行控制。Role {ACR，CAR} 其具体定义如下：

(1)针对数据的存取控制权限，包括：可读对象集：该角色可以读取的数据对象的集合；可写对象集：该角色可以修改的数据对象的集合；优先级：定义针对数据对象该角色的访问优先级，当访问发生冲突的时候可以依据优先级决定哪个用户可以获得数据的访问权。

(2)针对操作的协同感知权限。主要从两个方面描述，一是协同粒度，二是协同感知模式。协同粒度包括三种不同粒度的协同关系：

细粒度协同：将最细微的动作，如鼠标移动等设备的动作，通知协同的用户。

中粒度协同：用户完成有一定意义的动作后将动作结果通知协同的用户。

粗粒度协同：多用在 WYSINWIS 的协同系统中，协同用户只能感知动作的性质(如：用户 A 的鼠标被移动)而不能观察到动作的具体结果。

协同感知模式描述用户之间关系的密切程度：

完全感知模式：用户的全局感知行为集合，即该用户的行为协同系统中的用户都能感知。

部分感知模式：用户部分感知行为集合，只有在同一个组中的用户方可感知。

无感知模式：不可感知行为，用户的私有行为，不被其他用户感知。

综上所述，协同控制可以分为两个层次：一是整个协同工作过程的协同控制，一是参与协同的用户的协同描述。因此用户协同层模型主要分为组和角色两部分。由组来对一次协同工作中的用户之间的关系进行协同控制，而角色则用来确定用户和数据、设备之间访问关系。

4 UCL 的实现和应用实例

基于 LCSDM 模型我们设计和实现了一个支持 CSCW 开发的系统 JCSDK(Java-based Cooperative System Development Kit)。它建立在 Java 程序设计环境之上，用户协同层 UCL 即为其中的一个模块。

之所以选用 Java 作为实现语言，是因为我们的系统需要有较好的平台无关性和可移植性，以保证应用程序能在一个分布的异构的计算机网络上正常运行。Java 作为一种平台无关的面向对象语言对网络相关的技术带来了极大的影响，而且由于面向对象语言具有的封装、继承、重载等性质以及 Java 在安全方面的特性，以 Java 做为实现语言就可以更方便地实现扩充并保持相对的独立性，可以比较容易达到系统易移植性和安全性的目的。同时通过对角色和组的进一步分析，我们发现可以很方便地用类来实现这两者。角色和组都是系统中的对象，用户可以通过继承而获得相应的权限。

协同用户层 UCL 以 API 的形式提供给程序员。在其中建立了一些协同工作的用户所需遵循的协议和策略，并且由协同应用程序在一个协同工作的阶段贯彻。程序员只要在程序开始时加载 UCL 类库，就可以继承使用我们提供的类，从而较简单地实现用户协同的描述。用户协同层主要提供上面提到的几种类：

1) Group(组)：主要是对协同工作中的用户关系的描述。Group 是抽象类，在其中主要定义了一些组的基本属性和实现某些基本协同策略的方法。

对应于前面的三种协同模式，我们分别实现了三个相应的子类：EqualRelation，MasterSlave 和 ClientServer，并且在其中实现了简单的等待和并发控制策略。

2) Role(角色)：用户基本角色的实现。Role 为一抽象类，主要定义了一些基本属性和方法的名称。

依据对协同系统的分析，我们实现了几个基本的角色：Administrator，CoWorker，Watcher，Teacher，Student 等等，按照各种角色分别实现了相应的数据访问控制和协同感知控制。用户可以按需加入新的角色。

以虚拟实验室为背景，我们使用 JCSDK，实现了协同编辑和多用户讨论等几个实例，并且比较简单地实现了多粒度协同和在一个协同系统中多角色的定义。

协同编辑器 CoEditor 由一个文本编辑区(1)、讨

论区(2)和协同消息区(3)构成。用户在编辑区协同地编写文件并通过讨论区进行协商，在消息区用户可以获知其它用户的最新状态。工作组的类型为主从模式，其中包括三种用户角色：领导者，协同者和观察者。与此同时，协同编辑还要保证数据一致性、进行并发控制等。这些功能的实现如果采用集中控制的 C/S 结构，可以很方便实现操作的时序化[8]，从而比较容易保证数据一致进行并发控制。因此工作组的类型同时又是客户/服务器模式的。

● 领导者是协同编辑的发起人，拥有最大权限，可以提升和降低用户权限。

● 协同者参与协同编辑，除了不可对用户权限进行修改外，其它权限和领导者相同。

● 观察者不参与直接编辑，只在讨论区发表意见，可以向领导者申请加入编辑。

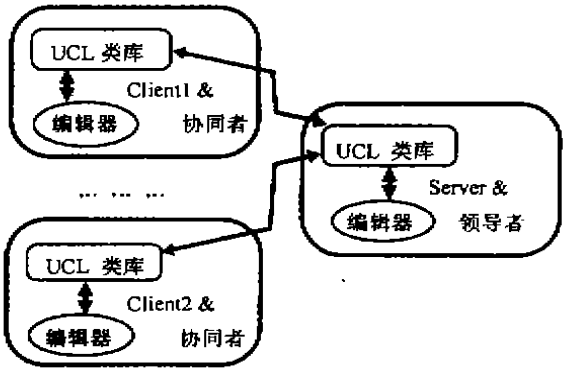

图5 CoEditor 中 UCL 类库的使用

CoEditor 程序和 UCL 类库直接交互，由类库完成所有底层的通信和协同工作，例如 Client1输入一段文字，当输入超过了其协同粒度时编辑器把输入转交 UCL 类库。由类库与 Server 处的 UCL 类库交互，将输入发送到 Server 处并修改某些协同信息。Server 的 UCL 类库进行一致性检查，若允许此操作则把 Client1的输入发到其它用户处。否则通知 Client1的 UCL 操作被拒绝，由它把编辑器的状态恢复到修改前的状态。

在例子程序中，我们还实现了多粒度协同，不同的用户角色对其它用户的行为的感知程度略有不同。在编辑器中我们提供了三种粒度：粗、中、细，用户可以自行选择希望的协同粒度以适应自己的编辑需求。

结束语 JCSDK 依据协同系统不同的构成方法，对其开发提供多个抽象层次的支持，尤其在用户协同层实现了高层的基于角色(role-based)的协同描述工具，用以描述协同系统的用户协同关系，从而更加方便用户开发协同的应用系统。当然，我们的系统还有待进一步完善，用户层就只实现了对象协同层的一部分功能，对 UCL 类库不支持的部分，程序员就必须使用对象层提供的较低级的支持。这将是我们进一步的工作。

参 考 文 献

1 Cortes M, Mishra P. DCWPL: A Programming Language For Describing Collaborative Work. ACM, 1996(11): 21～39

2 DePaoli F, Tiasato F. CSDL: A Language for cooperative systems design. IEEE Transactions on Software Engineering, 1994, 20(8)

3 Olson G, et al. Defining a metaphor for group work. IEEE Software, 1992(5): 93～95

4 Ellis C G, Simon J. Groupware some issues and experiences. Comm. of ACM, 1991, 34(1): 38～58

5 Grief I, et al. A Case Study Of CES: A Distributed Collaborative Editing System Implemented In Argus. IEEE T-SE, 1992(9). 827～839

6 Watson R, et al Culture: A fourth dimension of group support system. Comm. of ACM, 1994, 37(10): 44～55

7 Greenberg S, et al. Concurrency control in Groupware. In: Proc. of the CSCW'94. Chapel Hill NC. ACM Press, 1994. 207～217

8 Pendergast M O. A Comparative Analysis of Groupware Application Protocols. Computer Communication Review, 1998, 28(1): 28～41

9 茅兵，杜兴，谢立. 设计计算机辅助协同工作系统的几个关键技术. 计算机研究与发展，1996，33(4)：241～247

10 Li Chengkai, Mao Bing, Xie Li. LCSDM: A layered model for cooperative systems development. In: The Fifth Intel. Conf. for Young Computer Scientists(ICYCS'99)