计算机科学 1998Vol. 25№2. 6

GOM：一个 CSCW 设计模型

GOM-a General CSCW Design Model

刘厚泉

（中国矿业大学计算机科学与技术系 徐州 221008）

茅 兵 谢 立

（南京大学计算机科学与技术系 南京 210093）

摘 要 GOM is a general CSCW design model which divides CSCW system into user's personal space and sharing space. Each space consists of several objects, which are connected with information channels. This paper introduces the architecture of GOM, primarily focuses on functions of the information channel and it's specification

关键词 CSCW, Collaboration management, Specification language

1. 引言

自 Greif 和 Cashman 于 1984 年提出 CSCW（计算机辅助协同工作）概念以来，已有许多原型系统和商用系统问世。它们大致可分为如下几类[1]：(1)会议支持系统；(2)共享工作空间系统；(3)共享信息空间；(4)以计算机为媒介的通讯系统；(5)工作流和过程控制系统。

目前在 CSCW 的研究和开发中所存在的主要问题是缺乏支持协同系统和协同感知应用的一般化结构模型[2]，这一方面缘于 CSCW 涉及到诸如社会学、组织科学、心理学和计算机科学等广泛领域的问题[3]；另一方面，即使在同一个领域中，人们所需支持的强度也不尽相同。尽管如此，人们对具有一般意义的 CSCW 结构仍怀有浓厚的兴趣，并提出了若干方案[4~7]。

本文提出了一个具有一般意义的 CSCW 设计模型 GOM（Group Oriented Management），它将 CSCW 系统划分为各用户的私有空间和共享空间，并以"对象"概念来构筑各个空间。为了增强系统的灵活性和可重构性，GOM 的对象之间通过联接机制实现消息的传递。联接机制包括成员组对象和消息通道两个概念，前者用于管理各成员对象在协同工作中的角色，后者用于对象消息的分布、转换、检测等处理。

本文着重介绍了 GOM 的结构以及消息通道的功能和描述方法，而对另一些概念，如通道描述语言的语法和各类对象的结构形式等未作详细论述。

2. GOM 的结构

2.1 私有空间、共享空间和联接机制

根据用户与任务的关系，我们可将计算机处理环境分为四类：

(1)单用户单任务环境。一个用户以独占方式使用所有系统资源，并且每次只能处理一个任务。

(2)单用户多任务环境。一个用户可同时处理多个任务，且各任务之间存在信息的共享和交换。

(3)多用户多任务环境。传统的分布式处理即属于该类环境。它的特点是每个用户各自处理自己的任务，在通常情况下各用户无需关心其它用户任务的存在，而将整个计算机环境看作被自己所独占。

(4)多用户单任务环境。我们认为 CSCW 即属于该类环境。当然这里的单任务并非指整个系统中仅存在一个任务，而是指多个用户围绕着某一个任务开展协同工作。各用户不仅要关心自己对任务的处理情况，同时还要感知其它用户对任务的处理情况。

在这样的多用户单任务环境中存在着两类不同的空间，即用户的私有空间和用户共享空间。用户私有空间包括 GUI、数据管理等概念，用户可以通过它们感知和处理协同任务。每个用户私有空间相对独立，各用户能够以不同的方式和手段参与同一协同工作，并可重复利用单用户环境下的已有软件。由于每个用户各自的私有空间及其对协同任务关注的

侧面不尽相同，因此系统使用一个独立的共享空间的概念来描述协同任务的内涵。我们可以把共享空间看作一个抽象的空间，用户使用该空间的概念实现协同任务的定义、分解、及版本管理等工作。为了将不同空间的概念联接成一个整体，GOM 引入了联接机制的概念。联接机制在私有空间和共享空间之间建立映射，当某用户处理私有空间中与协同任务有关的问题时，联接机制将结果映射到共享空间中，从而引起共享空间抽象协同任务状态的改变；而当共享空间抽象协同任务改变时，联接机制又将变更结果映射到其它有关的私有空间(图 1)。

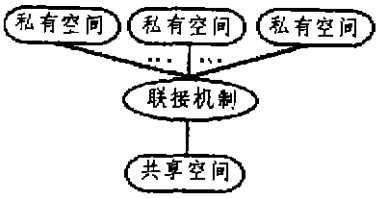

图 1　GOM 将 CSCW 环境划分为私有空间、共享空间及联接机制

2.2　GOM 的对象－关联结构

在 GOM 中，我们用成员对象来描述私有空间中与协同工作有关的概念，用设施对象来描述共享空间的概念。与私有空间和共享空间相比，联接机制具有更大的灵活性和复杂性，为此我们引入了成员组对象的概念，并在成员对象与成员组对象之间建立协同关联、在成员组对象与设施对象之间建立任务关联(图 2)。

图 2　GOM 的对象－关联结构

(1)成员对象：GOM 把每个用户私有空间中与协同任务有关的所有成分，如私有数据、GUI 等包裹起来，形成一个复合的对象，称为成员对象。它通过一个输入输出界面与其它对象联系。

(2)设施对象：是指在协同工作中可被多个成员对象所共享的对象，如公共数据、共享的 GUI 等。在 GOM 中，共享的 GUI 类对象实际上属于虚拟对象，即该类对象仅保存自己的状态，而其实际功能需靠其它相关的成员对象去完成。例如用户在发出修改某个共享窗口的消息时，该共享窗口对象将修改消息分布到相关的成员对象，并由各成员对象去修改自己的对应窗口。

(3)成员组对象：在协同工作过程中，各用户所担当的角色可能有所不同，所以各成员对象对设施对象的访问权限也不一样。如某些成员对象拥有修改共享数据的权限，而另一些成员对象仅拥有读取数据的权限。GOM 使用成员组对象来管理成员对象的权限。它一方面负责给成员对象访问设施对象的消息增加权限信息，另一方面负责将设施对象反馈来的消息分发给组内的有关成员。

(4)协同关联：在 GOM 的"对象－关联"结构中，所谓关联是指对象之间存在的消息传递关系。协同关联是指成员对象与成员组对象之间的关联。在某成员对象和某成员组对象建立了协同关联以后，成员对象发往设施对象的消息，以及设施对象的反馈消息都需经过成员组对象而转发。协同关联可在系统运行前静态地建立，也可在系统运行中动态地建立。在动态建立协同关联中，又可分为主动建立和被动建立两种方式。

(5)任务关联：即成员组对象与设施对象之间的关联。一般地说，对于确定的协同任务，其所允许关联的成员组对象是固定的，所以任务关联不会随成员对象角色的动态改变而改变。

3. GOM 的消息通道

3.1　消息通道的结构

由于各成员对象及设施对象分属于不同空间的概念，且各用户对协同任务的需求也不一样，所以在各类对象的消息传递中有必要对其进行检测、转换等处理，我们称之为消息的关联处理。GOM 为此设置了消息通道来负责消息的关联处理(图 3)。

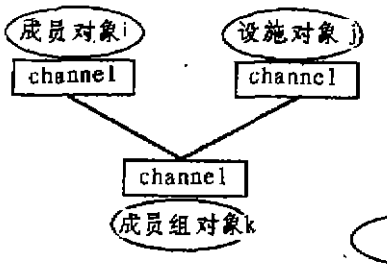

图 3

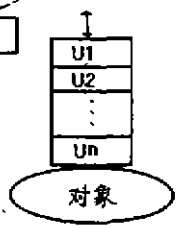

图 4

消息通道的功能如下：

(1)消息分布:将一消息有选择地传送给多个相关对象。如将共享对象的变更消息传送给有关的成员对象等。

(2)条件检测:检测消息传送的条件。如果条件不满足则拒绝消息的传递。

(3)消息转换:成员对象与设施对象分属于不同空间的概念,消息转换可支持各用户以不同的方式(WYSIWIS 或 WYSINWIS)、不同的媒体(声音或图像等)来参与同一个协同任务。

(4)权限检测:检测各成员组对象访问共享设施对象的条件。

(5)冲突控制:控制多个对象对同一个对象的访问冲突。

(6)置优先级:设置多个对象或多条消息在消息发送和接收时的优先级。

(7)同步控制:控制消息之间的同步关系。

(8)消息捆绑:在协同工作中,有时必须等多条消息都发出时,才能决定下一步的动作,我们把这些存在内在关联的消息称为原子性消息。原子性消息在一些 CSCW 系统中具有重要的作用,如在会议支持系统中,只有当听众都发出准备好的消息时,发言用户才可以开始讲话。

GOM 的消息通道分为多个单元,每个单元对特定的消息作特定的关联处理,我们称这种单元为消息关联处理单元,简称为处理单元(图 4)。

3.2 消息通道的描述

GOM 提供了对消息通道的描述语言,用户可使用它对消息通道功能作形式化定义、检查等,系统能够自动将其转换为目标语言代码。我们在描述消息通道时使用了几个有关集合的概念。

(1)消息集:若干消息的集合。

(2)标识集:可以是若干成员对象的标识集合,也可以是若干成员组对象的标识集合或若干设施对象的标识集合。

(3)条件集:若干条件表达式的集合。

需要强调的是,我们所使用的集合与普通的集合在成员关系的概念上有所不同。在消息集、标识集和条件集中,成员之间存在着有序、无序、与和或的关系。

(1)有序集:成员之间存在着从左至右的顺序关系。我们将有序集表示为(m1,m2,…,mn)的形式。

(2)无序集:成员之间无顺序关系。我们将无序集表示为(m1,m2,…mn)的形式。

(3)与集:成员之间存在"与"的关系,我们用"、"符号分割与集的成员,如(m1、m2、m3、m4)为无序与集。

(4)或集:成员之间存在"或"的关系。我们用","符号分割或集的成员,如〈m1,m2,m3,m4〉为有序或集。

集合元素之间的有序、无序、与和或的关系对不同性质的集合具有不同的含义,例如对消息集来说,有序和无序表达了在发送或接收这些消息时是否有次序的要求,而与和或的关系则表达了这些消息是否必须同时处理;对标识集来说,有序和无序表达了这些对象之间是否存在优先级的问题;对条件集来说,与和或关系表达了这些条件是否需要同时满足。

由于消息通道由处理单元构成,所以我们首先给出消息关联处理单元的描述形式。

```
PUNIT u_name{
   INPUT:
      ADDR_SET=addr_set;
      MESS_SET=mess_set;
   CONDITION:
      COND_SET=cond_set;
   OUTPUT:
      ADDR_SET=addr_set;
      MESS_SET=mess_set;
}
```

其中 u_name 表示处理单元名。单元体可分为三部分,INPUT 部分描述了可进入处理单元的对象标识集和消息集;OUTPUT 部分描述了经处理后形成的标识集和消息集。而当 CONDITION 部分的条件集为假值时,INPUT 部分的消息将被拒绝通过消息通道。

GOM 的消息通道由一个消息关联处理单元的集合构成,即:

```
CHANNEL ch_name{
    PUNIT_SET=punit_set;
}
```

消息通道的功能主要通过处理单元的功能来体现,以下举例说明之。

(1)消息分布

```
/*将本对象的两条消息 message1和 message2
   顺序发送给对象 object1,object2*/
PUNIT u_name{
    INPUT:
       MESS_SET=(message1,message2);
    OUTPUT:
       ADDR_SET=(object1,object2);
}
```

(2)消息捆绑

```
/*当对象 object1,object2同时发出消息 message1时,将其
转换为两条顺序消息后传送给本对象*/
PUNIT u_name{
 INPUT:
  ADDR_SET=(object1,object2);
  MESS_SET=(message1);
 OUTPUT:
```

```
    MESS_SET=(message2,message3);
;
```

(3)消息转换

```
/*将从对象object1发来的消息message1转换为两条顺序
消息message2,message3后传给本对象*/
PUNIT u_name{
 INPUT,
    ADDR_SET=(object1),
    MESS_SET=(message1);
 OUTPUT;
   MESS_SET=(message2,message3);
}
```

(4)条件检测

```
/*对于所有其它对象发来的消息message1,
当条件condition1及condition2同时满足时传给本对象*/
PUNIT u_name{
 INPUT;
    ADDR_SET=ALL;
    MESS_SET=(message1);
 CONDITION,
    COND_SET=(condition1,condition2);
;
```

(5)置优先级

```
/*设置对象object1,object2,object3对本对象访问的优先
级*/
PUNIT u_name{
 INPUT;
    ADDR_SET=(object1,object2,object3);
    MESS_SET=ALL;
```

4. 消息通道的描述实例

我们以一个"研讨会"形式的CSCW应用系统实例来说明GOM消息通道的描述。图5为该CSCW应用系统的结构图。

在此我们不详细叙述各对象的功能和结构,但为了方便对消息通道的描述,仅作以下说明:

·发言席对象和提问席对象是系统的两个共享设施对象,各成员对象通过与发言席(提问席)对象的通信而实现发言(提问)。

·在每个成员对象中有两个图文编辑/显示窗口,分别对应共享空间中的发言席对象和提问席对象。当某一对象在其私有的发言(提问)窗口中输入信息时,其内容便会在其它相关成员的发言(提问)窗口中显示出来。

·发言组对象、提问组对象和听众组对象各自维护着一张属于该组的成员的列表,并根据该列表决定各成员的角色。

以下我们给出系统中三个通道的形式化描述。

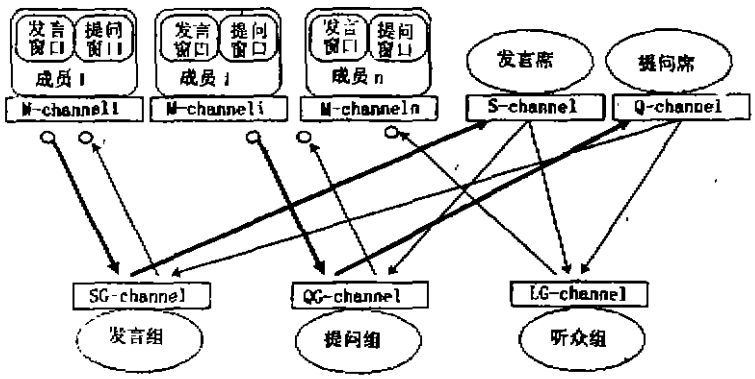

图5 "研讨会"系统的结构图

(1)成员对象通道的描述。由于各成员之间采用WYSIWIS的交互方式,故成员对象通道无需对输入和输出的消息作任何处理。

```
CHANNEL M_channeli{
    PUNIT_SET=NULL;
}
```

(2)发言组对象通道的描述。根据"研讨会"任务的性质,我们认为该通道应具有如下功能:

·当发言组中没有任何成员时,才允许成员加入该组;

·当组中有一个成员,且与申请退出该组的成员标识相符时,才允许退出该组的消息通过;

·将发言组对象的发言消息送往发言席对象;

·将提问席对象的提问消息送往发言组对象。

我们将以上通道的四项功能描述为如下的处理单元:

```
PUNIT spk_grop1{
    INPUT;
       ADDR_SET=ALL;
       MESS_SET=(join);
    COND_SET=(#Spk_Grop=0);
}
PUNIT spk_grop2{
    INPUT;
```

```
        ADDR_SET=ALL;
        MESS_SET=(leave);
    COND_SET=(#Spk_Grop=1,
    ($Spk_Grop)=(leave.memb));
}
PUNIT spk_grop3{
    INPUT:
        ADDR_SET=($Spk_Grop);
        MESS_SET=(speak);
    OUTPUT:
        ADDR_SET=(Speaker);
}
PUNIT spk_grop4{
    INPUT:
        ADDR_SET=(Questioner);
        MESS_SET=(question);
    OUTPUT:
        ADDR_SET=($Spk_Grop);
}
```

发言组对象通道的描述形式为

```
CHANNEL SG_channel{
    PUNIT_SET=(spk_grop1,spk_grop2,spk_grop3,
    spk_grop4);
}
```

(3)发言席对象通道的描述。发言席对象应具有的功能为：

·将发言组对象的发言消息传送给提问席对象和听众席对象。

处理单元的描述为

```
PUNIT speaker1{
    INPUT:
        ADDR_SET=(Spk_Grop);
        MESS_SET=(speak);
    OUTPUT:
        ADDR_SET=(Questioner,Listener);
}
```

发言席对象通道的描述形式为

```
CHANNEL S_channel{
    PUNIT_SET=(speaker1);
}
```

结束语 作为一个具有一般意义的 CSCW 设计模型，GOM 具有如下特点：

(1)在结构上将 CSCW 系统划分为用户私有空间、共享空间和联接机制三部分。用户使用共享空间的概念来定义协同任务，而使用私有空间的概念来感知和处理协同任务。它保证了协同任务与各用户环境的无关性，使得用户能以不同的方法和手段来处理协同任务。

(2)由成员组对象统一管理各成员对象的角色，并与私有空间对象及共享空间对象建立协同关联和任务关联。它使得成员对象动态加入和离开某一成员组时不会引起设施对象消息传递的重定位。

(3)消息通道的语言描述使得其功能定义、检测十分方便，并可由系统将其转换为目标语言代码。与文[2]的 CSDL 相比，消息通道描述语言不仅可以定义对象间的协同关系，还可以实现对象消息的转换、过滤等功能。

参 考 文 献

[1] M. S. Ackerman et al, Socila activity indicators for groupware, IEEE Computer, 26(6), 1996

[2] F. DePaoli et al, CSDL: A language for cooperative systems design, IEEE Trans. on Software Eng. 20(8), 1994

[3] L. Navarro et al, CSCW requires open systems, Computer Comm. 16(5), 1993

[4] K. Frooqui et al, Group communication models, Computer Comm. 19, 1996, 1276-1288

[5] 茅兵，协同系统软件结构的研究[博士论文]，南京大学. 1996

[6] 黄文倩，茅兵，谢立，KCHCI：一个 CSCW 系统框架，计算机研究与发展，19(12)，1996

[7] T. P. Liang et al, When client/server isn't enough: Coordinating multiple distributed tasks. IEEE Computer, 20(8), 1994