一种支持动态任务分配的协同设计方法*

刘 弘[1], 林宗楷[2]

[1](山东师范大学 计算机科学系,山东 济南 250014);

[2](中国科学院 计算技术研究所 CAD开放研究实验室,北京 100080)

E-mail: lhsdcn@pub.online.jn.sd.cn

http://www.xinx.sdnu.edu.cn

摘要: 协同设计是一种复杂的、由不同领域的专业人员参加的群体活动.介绍了一种多Agent设计系统中的协作方法,提出了支持动态任务分配的公告板机制以及设计过程中的冲突协调方法.公告板模型结合了黑板模型与合同网模型的优点,有效地利用Agent的自主性和协作性,采用分布与集中相结合的方式,克服了在分布式环境下进行动态任务分配时采用黑板模型与合同网模型的不足之处.同时,所提出的冲突预检查及冲突处理的方法可以在设计的早期阶段发现和解决冲突,以避免不必要的资源浪费.

关键词: CSCW;Agent;公告板;冲突协调

中图法分类号: TP311 **文献标识码**: A

随着计算机技术的发展,在互联网支持的计算平台上以计算机作为工具进行协同设计已成为设计界的发展趋势.设计是一个复杂的逐步求精的过程,在多Agent设计系统(multi-agent design system,简称MADS)中,复杂的设计由计算机软件(agent)与人类决策者共同完成,Agent作为设计者的助手,在整个设计生命周期中为设计者提供支持.多Agent技术为协同工作的支撑环境提供了框架结构及必需的底层支持,并且为Agent之间的信息共享与数据共享建立了纽带.

目前,国内外很多MADS的研究集中在Agent之间的信息共享与数据共享[1].但是,多Agent在设计过程中动态协作的研究对产生高质量的设计及有效的资源利用是同样重要的[2].

本文介绍了一种支持动态任务分配的公告板机制及设计过程中的冲突协调方法.公告板模型结合了黑板模型及合同网模型的优点,有效地利用Agent的自主性、协作性,采用分布与集中相结合的方式,克服了在分布式环境下进行动态任务分配时采用黑板模型与合同网模型的不足之处.同时,本文提出了一种将设计目标分解为设计目标树,并在协同设计过程中按设计目标在于设计目标树中自顶向下逐层进行冲突预检查及冲突处理的方法.该方法旨在设计的早期阶段发现和解决冲突,以避免不必要的资源浪费.

1 支持动态任务分配的公告板模型

黑板模型是协作时常采用的方法[3~5].该模型已经被广泛使用,并且其相应的技术已较为成熟,但仍存在以下问题:

• 在黑板模型中,由于公共数据库的存在而表现出一种强耦合性.虽然数据的集中存放给数据

* 收稿日期: 2000-04-05; 修改日期: 2000-08-24

基金项目: 国家自然科学基金资助项目(69975010);山东省自然科学基金资助项目(Q99G07);香港理工大学基金资助项目(S901)

作者简介: 刘弘(1955-),女,山东济南人,博士,教授,主要研究领域为多Agent系统,CSCW;林宗楷(1934-),男,福建莆田人,研究员,博士生导师,主要研究领域为CSCW,CAD,工程数据库.

的一致性带来了许多益处,但是因为在求解的中间过程产生的数据都要传到黑板上,所以这也成为系统的一个瓶颈.

·各知识源通过黑板进行交互时必须转换为规定的格式,这给具有不同知识以及使用不同工具的 Agent 的协作造成了一定的困难.

在开放的分布式环境中,任务和各 Agent 的处理能力都在不断改变.由于合同网(contract-net)存在的动态特性,常使用这种协作组织模型[6,7].

在合同网方式下,要求得到服务的 Agent 以广播方式进行招标,可以提供服务的 Agent 向招标的 Agent 发"bids"进行投标,招标 Agent 根据对"bids"的评价选择中标 Agent,并与其订立合同进行协作.

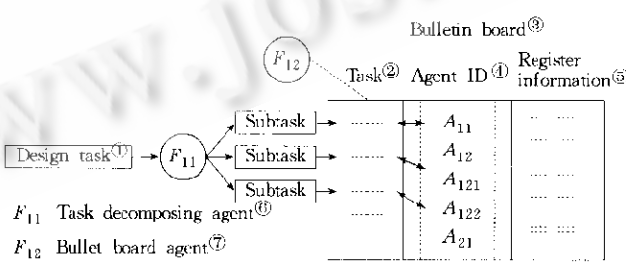
合同网模型中主要存在两方面的问题:任务分解和任务分配.当请求者分解任务时,任务固有的分解方式可能不适合于开放的分布式环境.请求者不知道当时可以提供怎样的 Agent,也不知道潜在的被请求者有哪些技能.因此,他没有足够的知识正确地分解一个复杂的目标,单个 Agent 又没有足够的能力去独立完成复杂的任务,从而造成可以由多个 Agent 协作完成的任务没有承担者.

同时,由于多 Agent 系统中 Agent 的粒度和规划在动态地加以改变,Agent 的类型、地位和权限也不相同,这些附加的复杂性使得多 Agent 系统中应用合同网非常困难.

针对上述问题,我们结合黑板模型及合同网模型的优点,提出了一种公告板模型.

在公告板方式下,各 Agent 有自己的知识源,能以不同的格式存放在不同的地点.当想要参加协作时,只需将自己的名字、物理位置、资源、当前状态、如何与他们联系、可以提供的服务或需要得到的服务在公告板上登记,即可由公告板 Agent 辅助建立 Agent 之间匿名的互操作和知识共享.公告板 Agent 匹配任务及 Agent,把适当的任务及 Agent 之间进行联系所需的信息传送给适当的 Agent,为 Agent 之间进行联系提供路由服务.与任务有关的 Agent 可以通过公告板建立联系,进行点对点的通信.Agent 本身并不知道与己无关的任务由哪些 Agent 来完成,并且由于多数资源分布在不同地点,减少了对共享库频繁访问的通信资源及繁琐的数据转换,使通信带宽得到充分的利用.

一个公告板的例子如图 1 所示.



①设计任务,②任务,③公告板,④Agent标识符,⑤注册信息,⑥任务分解Agent,⑦公告板Agent.

Fig. 1 An example of bulletin board

图1 一个公告板的例子

·注册.需要提供服务或得到服务的 Agent 在公告板上注册其名字、物理位置、资源、当前状态、如何与他们联系以及可以提供的服务或者将想要得到的服务在公告板上登记.在 Agent 离开系统时,要向公告板发送取消注册信息(这也体现了 Agent 的自主性与协作性).公告板 Agent 为每一个注册的 Agent 分配一个标识符,并且记录各 Agent 的注册信息和任务表,即公告板始终保

持着当前分布式环境中活跃的 Agent 的路由信息.

注册表的形式如下:

Name:	E-mail	Address:
Location:	www	Address:
Resource:	Offer	service:
Status:		

· 匹配任务和 Agent. 任务分解 Agent 根据注册 Agent 可以提供的服务、资源和状态,对请求得到的服务(任务)按申请顺序进行分解,公告板 Agent 根据任务的特征和 Agent 的注册信息匹配任务和 Agent,并在 Agent 同意接受任务以后,把 Agent 之间进行联系所需的信息传送给与任务有关的 Agent.

任务分解以及与相应的 Agent 匹配的工作过程如下:

(1) 任务分解

当任务分解 Agent F_{11} 接受了设计任务 t_j 时,它执行以下操作:

(a) 添加 t_j 到其任务表 T 中;

(b) 当 t_j 在 T 的任务序列中到达队首时,F_{11} 从 T 中取出 t_j,并在任务分解库中查找可能的分解方案,如果 F_{11} 曾经执行过类似的分解任务,库中将可以查到 t_j 的设计目标树(见第 2 节的定义 2),如果查不到类似的历史记录,F_{11} 会请求设计工程师进行任务分解,并把设计工程师的分解方案存入任务分解库;

(c) F_{11} 将分解结果传给公告板 Agent F_{12}.

(2) 任务分配

F_{12} 根据下列原则将设计任务与 Agent 进行匹配:

(a) 如果有多个适合完成任务且任务表为空的 Agent,则按注册顺序选择;

(b) 如果合适的 Agent 都在执行任务,则按以下原则:

· 选择任务较少的;

· 选择可信度较高的.

公告板记录了每个曾经注册的 Agent 所完成任务的次数及情况,并根据次数及完成情况计算 Agent 的可信度. 计算公式为

$$Reliability = W_1 * Times + W_2 * Value$$

其中,$Reliability$ 为可信度,$Times$ 为完成任务的次数,$Value$ 为完成任务情况的评价值,W_1 与 W_2 分别为次数及评价的权值,可由专家根据具体任务来调整;

(c) 使协作组内的成员之间的物理距离较短.

考虑到资源共享、数据传输的花费及协作的方便,尽量使协作完成同一个任务的 Agent 之间的物理距离较短.

如果某项任务找不到合适的 Agent,则将其返回任务分解 Agent 作进一步分解,或请求专家帮助,并将任务分配情况添加到设计目标树中.

· 协作. 当任务和 Agent 被匹配以后,各 Agent 根据公告板 Agent 提供的信息进行联系,各 Agent 所接受的设计任务在目标树上的位置确定了他们之间的协作关系. 任一 Agent 完成当前的任务之后,都将当前状态通知公告板 agent,以便公告板 agent 修改注册信息以及对该 Agent 的评价信息,并重新分配任务.

2 支持协作的设计事务模型

在协同设计中,由于参与设计的用户和 Agent 有不同的背景和设计方案,不能简单地把多个子设计合并起来,就形成一个整体设计.因此,在将任务分解并分配给多个设计人员及相应的设计 Agent 以后,各设计 Agent 除了要独立完成设计任务以外,还要有全局观念,即当个体的设计与全局设计发生冲突时,要根据整体的设计方案调整自己的局部设计方案.本节提出了一种提供保存、回退机制的设计事务模型,以支持设计过程中设计方案的协调及相应的修改.

该事务模型的主要思想是,当设计 Agent(通过与设计者交互获得该知识)认为某一时刻的设计状态有保留价值时(以后可能回退到这里),则在此处设立一个保存点.设计工作空间的主体是由设计状态(保存点)组成的一棵树.保存点具有 3 个重要性质:(1) 保存点是可以回退的,即从当前工作状态可以回退到先前设定的任意一个保存点;(2) 保存点是稳定的,即每个保存点都是可以恢复的;(3) 保存点是可以跨越多个进程的.

从最近的一个保存点开始的一段设计过程构成了设计者的当前工作路径.当前工作路径是由不同的状态构成的线性序列,序列中从前一个状态到后一个状态的转移是由一个对象操作引起的.设计者可以通过取消(undo)和重做(redo)操作在当前工作路径上作线性搜索以便寻求一个有用的设计点.

设计事务模型形式定义如下:

定义 1. 一个设计目标可以表示成一个四元组(G,D,A,FG).其中 G 为目标名,D 表示设计内容,包括各项约束条件及要求达到的技术指标,A 是负责对 G 进行设计的 Agent 的名字,FG 表示该设计目标的父目标.

定义 2. 一棵设计目标树(简称目标树)是以总设计目标为根结点,按照一定的分解关系逐层分解,由设计目标组成的一棵树.

定义 3. 设计方案树(简称方案树)是以某一设计目标为根结点,以实现该目标的方案为子结点的与/或树."与"结点表示大目标分解为若干个小目标;"或"结点表示该目标的多个实现方案.

定义 4. 设计方案空间(简称方案空间)是一棵设计目标树对应的所有设计方案的集合.

定义 5. 设计工作空间是一个四元组:$(SavepointTree, WorkingPath, CurrentWorkingScheme, Status)$,其中:

(1) $SavepointTree$ 是一个三元组:(S_0,N,B),代表一棵多叉树,其中 $S_0 \in SchemeSpace$,是树的根;$N \subset SchemeSpace$,是树的结点集;B 是二元组(S_i,S_j)的集合,(S_i,S_j)表示树中一条从父结点 S_i 到子结点 S_j 的分枝,$S_i,S_j \in N$.

(2) $WorkingPath$ 是一个序列:$(S_{i0},S_{i1},S_{i2},\ldots,S_{in})$,其中 $S_{ij}(j=1,2,\ldots,n) \in SchemeSpace$,$S_{ic} \in N$.

(3) $CurrentWorkingScheme$ 是 $WorkingPath$ 中的一个成员.

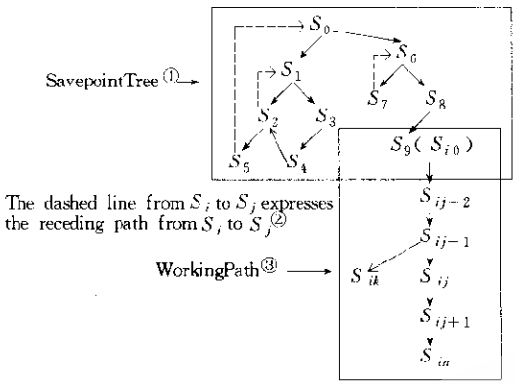
(4) $Status$ 的值域是集合{ACTIVE,DEACTIVE,PAUSE}.

ACTIVE,DEACTIVE,PAUSE 分别表示设计事务处于活动、完成和暂停状态(如图 2 所示).

定义 6. 设计状态转换(transfer)操作将满足某一条件(condition)的一个设计状态(Design.state1)转换为另一个状态(Design.state2).

Transfer ($Design.state1 \rightarrow Design.state2$) While Condition

定义 7. 设计状态转换操作空间 DSTS(design state transfer space)是所有设计状态转换操作



①保存点树,②从S_i到S_j的有向虚线表示从S_i对应的工作路径回溯到S_j,③工作路径.

Fig. 2 An example of design affair model

图2 一个设计事务模型的例子

的集合,由 *OPName* 和 *OP* 组成. 其中

OPName = {*transaction. begin*, *transaction. commit*, *transaction. abort*, *savepoint*, *rollback*, *undo*, *redo*, *pause*, *resume*}

OP = {*opi* (*scheme*1 → *scheme*2), *opi* ∈ *OPName*, *scheme*1, *scheme*2 ∈ *SchemeSpace*, $i = 1, 2, \ldots, 9$}, 即 *OP* 为对应于 *OPName* 中操作名 *opi* 的状态转换操作序列.

定义 8. 设计事务是一个以 *transaction. begin* 开始,以 *transaction. commit* 或 *transaction. abort* 结束,中间由多个设计状态转换操作构成的序列.

设计状态转换操作 savepoint, rollback 支持设计过程中的保存/回退, undo, redo 反映了当前工作路径上的局部试探行为, pause, resume 体现了一段设计可以跨越多个进程.

3 设计方案冲突的协调解决方法

本节介绍在协同设计过程中,按照设计目标在设计目标树中自顶向下逐层进行冲突预检查及冲突处理的方法.

每个设计 Agent 在进行设计时,都先根据确定的目标,制定尽可能多的满足目标要求的方案,然后对各方案进行可行性估计,确定方案的可行性估值,形成自己的方案空间,以便在冲突消解过程中进行有效的替换.

对于具体方案的可行性估值不仅因人(不同领域的设计 Agent)而异,而且因目标的不同而不同,这就增大了给出方案可行性估值算法的难度. 由于对每一个 Agent 而言,它对一个方案进行可行性估值不仅是为了看其是否可行,而且更重要的是为了在满足同一目标的方案中进行比较,从中选出较好的方案,因此,单个 Agent 在进行独立设计时,对同一个目标的多个方案进行可行性估值所依据的原则是一致的.

当然,不同的设计 Agent 对不同目标的设计方案的估值方法并不相同. 因此,应由一个 Agent 从全局的角度考虑,按统一的标准进行可行性估值. 下面,我们给出确定可行性估值的一般性方法.

设有 n 个属性 $D_1, D_2, \ldots, D_n$, 可用来刻画方案 P 的可行性,并设这 n 个属性之值分别为 $x_1, x_2, \ldots, x_n$. 各属性的重要性不尽相同,我们用 K_i 来表示属性 D_i 的权值,K_i 由进行可行性估计的专家 Agent 给出,则对设计方案 P 的可行性估值为

$$M(P) = u(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} K_i * u_i(x_i), \quad (1)$$

其中 $u_i(x) = |x - x_i w| / |x_i b - x_i w|$.

$x_i b$ 表示属性 D_i 的最佳属性值,$x_i w$ 表示属性 D_i 的最差属性值.

我们假定存在属性 D_i 的最佳属性值与最差属性值,例如,在进行某项设计时,要求至少满足用户要求的 90%,则 $x_i b = 1$, $x_i w = 0.9$.

由 $u_i(x)$ 的定义可知, $u_i(x_i b) = 1$, $u_i(x_i w) = 0$, $u_i(x)$ 在区间[0,1]取值.

定义 9. 设 A_1 与 A_2 为两个设计 Agent,分别对目标 G_1 和 G_2 进行设计,设 P_1 为 A_1 相对于 G_1 的候选方案,P_2 为 A_2 相对于 G_2 的候选方案,P_1 与 P_2 不相冲突,则称(P_1,P_2)为 A_1 与 A_2 的可调解方案.

定义 10. A_1 与 A_2 的所有可调解方案的集合叫做 A_1 与 A_2 的可调解方案集,记作 Mediacy (A_1,A_2).

方案折衷处理的思想是,当 A_1 与 A_2 分别从自己的角度选择的最佳设计方案发生冲突时,综合考虑各 Agent 的利益,采用选择可调解方案集 $Mediacy(A_1,A_2)$中两个方案的可行性估值之积的最大者作为最佳调解方案,即

$$\mathrm{MAX}(M(P_i)*M(P_j)) \qquad (P_i,P_j)\in Mediacy(A_1,A_2).$$

当 $Mediacy(A_1,A_2)$为空时,则需要与用户协商进行修改.

由于方案修改有可能引起连锁反应,因此,我们希望在设计的早期阶段发现冲突.针对这个问题,我们采用了按设计目标逐层进行冲突检查和冲突处理的算法.

该算法的基本思想是各设计 Agent 根据目标树确定各自的设计目标以及相互的联系.在分别形成自己的设计方案之后,按设计目标在目标树中的层次自顶向下地进行冲突检查.当各 Agent 针对上层设计目标提出的方案没有冲突后,再对下层设计目标按父结点分组,进行设计方案的协调.如果同组结点之间的设计方案是不可协调的,则要调整父结点的设计方案,并使整个协调过程回退一层.

算法.

Step 1. 初始化

对设计目标按其在目标树中的层次自上而下、自左至右进行排队(不包含根结点),队列中各元素的形式为(G,D,A,FG)(见定义 1).将属于同一个父结点的子设计目标分为一组,并从队列中取出第 1 组元素.

Step 2. 冲突检查

根据该组设计目标对应的设计方案进行冲突检查,如果无冲突,则从队列中取下一组,如果队列为空,转 Step 5.

Step 3. 冲突调解

对于有冲突的设计方案,按方案折衷的方法进行调解;对于组内无法进行调解的方案,回退至父结点.

Step 4. 回退调整、检查

相应的 Agent 根据子目标冲突的原因在其设计方案树中进行选择及调整,并且将调整后的方案与同组的设计方案重新做一次冲突检查.如果有冲突,转 Step 3;否则,转 Step 2,重新进行子结点组的检查.

Step 5. 结束冲突检查.

4 结束语

我们参与了香港理工大学基金项目,并已将本文提出的公告板模型以及协同设计过程中的冲突协调方法成功地应用于该项目,取得了良好的效果.目前,系统正在进一步完善及推广应用过程中.由于篇幅所限,应用实例略[11].

References:

[1] Liu, Hong. The study on construction approach of agent-based cooperative design system [Ph.D. Thesis]. Beijing: Institute of Computing Technology, the Chinese Academy of Science, 1998 (in Chinese).

[2] Liu, Hong, Zeng, Guang-zhou. A mechanical design oriented construction approach of agent system. Journal of Software, 1998,9(6):88~93 (in Chinese).

[3] Hayes-Roth, B. BBI: an architecture for blackboard systems that control, explain, and learn about their own behavior. Heuristic Programming Project Report, HP-84-16, Stanford, CA: Stanford University, 1984.

[4] Nii, H.P. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. AI Magazine, 1986,7(2):38~53.

[5] Nii, H.P. Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. AI Magazine, 1986,7(3):82~106.

[6] Davis, R., Smith, R. Negotiation as a metaphor for distributed problem solving. Artificial Intelligence, 1983,20(1):63~109.

[7] Smith, R. The contract-net protocol: high level communication and control in a distributed problem solver. IEEE Transactions on Computers, 1980,29(12):1104~1113.

附中文参考文献:

[1] 刘弘.基于 Agent 的协同设计系统构造方法研究[博士学位论文].北京:中国科学院计算技术研究所,1998.

[2] 刘弘,曾广周.面向机械设计的 Agent 系统构造方法.软件学报,1998,9(6):89~93.

A Cooperative Design Approach Supporting Dynamic Task Assignation*

LIU Hong[1], LIN Zong-kai[2]

[1](*Department of Computer Science, Shandong Normal University, Ji'nan* 250014, *China*);

[2](*CAD Laboratory, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: lhsdcn@pub.online.jn.sd.cn

http://www.xinx.sdnu.edu.cn

Abstract: Cooperative design is a complex group activity that involves participants with heterogeneous skill. In this paper, a cooperative design approach in a multi-agent design system is introduced and a form of Bulletin Board that supports dynamic task assignation and a kind of conciliatory approach when conflict occurs in design are presented. According to autonomous and cooperative features, the form of Bulletin Board combines the advantage of black board and contract-net and introduces a style of distributed and centralized combination, weakens some failings of black board and contract-net during dynamic assignation in distributed environment. Then, it puts forward a conflict pre-checking and conflict processing approach. This approach aims at finding and solving conflict during the early design phase and decreasing resource waste.

Key words: CSCW; agent; bulletin board; conflict conciliation

* Received April 5, 2000; accepted August 24, 2000

Supported by the National Natural Science Foundation of China under Grant No. 69975010; the Natural Science Foundation of Shandong Province of China under Grant No. Q99G07; the Hong Kong Polytechnic University of China under Grant No. S901