# 层次结构文档下支持权限管理的实时协同技术研究*

高丽萍[1,2]，陈庆奎[1]，卢　暾[3]，高丽丽[4]

(1. 上海理工大学 光电信息与计算机工程学院，上海 200093；2. 上海市计算机软件评测重点实验室，上海 201114；3. 复旦大学 计算机科学技术学院，上海 200433；4. 潍坊学院 幼教特教师范学院，山东 潍坊 261021)

**摘　要**：针对传统实时协同系统中各站点之间无约束操作的缺点，提出采用权限分工的方式来支持不同团队之间的文档管理。给出了支持权限管理的层次结构文档的形式化定义，描述了用户角色设置过程，定义了权限及权限操作格式，讨论了站点 ID 的设置及初始化过程，并提出采用改进的地址空间转换算法来保证权限操作在各分布式站点的正确执行。分析了改进算法的效率提升，并通过其在 Co-AutoCAD 中的应用证实了策略的有效性。

**关键词**：协同设计；权限管理；实时协同；层次结构文档；地址空间转换

**中图分类号**：TP311　　**文献标志码**：A　　**文章编号**：1001-3695(2012)05-1690-05

doi:10.3969/j.issn.1001-3695.2012.05.023

# Research on real time collaboration technique supporting right management of layer-based document

GAO Li-ping[1,2], CHEN Qing-kui[1], LU Tun[3], GAO Li-li[4]

(1. *School of Optical-Electrical & Computer Engineering, University of Shanghai for Science & Technology, Shanghai* 200093, *China*; 2. *Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai* 201114, *China*; 3. *School of Computer Science, Fudan University, Shanghai* 200433, *China*; 4. *Pre-school & Special Education Normal School, Weifang University, Weifang Shandong* 261021, *China*)

**Abstract**: As for the lack of the constraint on operations in traditional real-time collaborative environment, this paper introduced adopting right division technique to support the document management of the shared document of different work groups. The paper gave formal definition of the layer document supporting role management, described the configuration process of the roles, defined the formats of the right operations, and discussed the set up of the site IDs and the initialization process of them. Beside, the paper improved the address space transformation strategy to guarantee the correct execution of the right operations in the distributed sites. Efficiency analysis of the improved algorithm is given and the application of this strategy in the Co-AutoCAD system proves the validity of the whole strategy.

**Key words**: collaborative design; right management; real-time collaboration; layered structure document; address space transformation

在 CAD 领域中，用户已经习惯于使用各种类型的编辑工具(如 AutoCAD、3DMax、Solid Work、Inventor 等)来将构思及草图转换为实体、对象和模型。但是这些工具无一例外都会碰到多个用户编辑同一个文档或者不同领域的用户使用不同的工具来编辑同一个产品的不同部分的问题[1]，这就对共享文档群编辑提出了新的要求。传统的协同工具(如文件共享、NetMeeting、IM、e-mail、版本控制等)[2,3]依靠设计人员自身来解决由于设计约束存在而产生的冲突，仅能提供异步协同和部分粗粒度的分模块协同的支持，存在诸如响应时间差、并发度低、冲突解决困难等缺点，对细粒度的实时交流和共享的支持非常有限。

实时协同采用复制式架构在各分布式站点之间构建 P2P 连接[4~7]。在全复制式架构中，每个协同站点上都有一份共享数据对象的文档副本。操作的本地站点产生后，会立即在其本地文档副本上得到执行，操作效果直接反馈给用户，从而确保群件系统具有与单用户应用程序同等的本地操作响应速度。实时协同系统通常采用操作转换(operation transformation, OT)[5,7]或地址空间转换(address space transformation, AST)[6]等乐观并发控制技术保证各分布式站点之间的文档一致性。由于无须采用加锁或令牌等悲观机制，实时协同具有高度的并发度，并具有无限制协同的特征。但将该架构用于协同设计环境来支持不同设计成员之间的协作与交流时，会产生如下问题：a)由于每个用户都拥有对共享文档的对等编辑权限，会导致不同团队之间用户的操作相互干扰(如管道设计人员对结构设计作修改)；b)将文档影射成线性文档，并在整个文档上执行并发控制会增加一致性维护算法的复杂程度；c)无法提供对于不公开文档的保密支持。

## 1　相关工作

权限管理是信息系统、网络安全等领域的研究热点。近年

**收稿日期**：2011-10-13；**修回日期**：2011-11-28　　**基金项目**：国家自然科学基金资助项目(60970012)；上海市教委“晨光计划”项目(10CG49)；上海高校选拔培养优秀青年教师科研专项基金资助项目(SLG10007)；上海理工大学光电学院创新基金建设资助项目(GDCX-Y1109)

**作者简介**：高丽萍(1980-)，女，山东莱阳人，讲师，博士，主要研究方向为 CSCW、一致性维护、协同设计等(lipinggao@fudan.edu.cn)；陈庆奎(1966-)，男，教授，硕士，主要研究方向为工作流协同、并行计算等；卢暾(1977-)，男(壮族)，讲师，博士，主要研究方向为分布式协同计算；高丽丽(1982-)，女，山东烟台人，助教，硕士，主要研究方向为 CSCW、进化计算等.

来,国内外学者就该领域中的访问控制策略(RBAC、TBAC等)[8,9]、支撑技术(基于 Hibernate、基于 SOA 等)[10,11]、通用架构[12]等方面作了大量研究,但这些研究大多是基于 C/S 或 B/S 架构的,其权限的分配通常由一个或多个超级用户来完成,无法满足复制式架构中对操作快速响应的要求及协同系统中对细粒度实体的访问控制需求。

协同环境下权限管理的研究较少。方萃浩等人[13]提出一个针对协同环境下 CAD 模型的多层次动态安全访问控制模型,该模型利用一种多层次的权限模型,以简化权限定义及其分配过程,丰富了权限表达能力,实现了产品模型的多粒度访问控制。然而该工作的重点在于解决工作流协同环境下产品模型之间在上下流传递过程中的权限分配及动态迁移问题。付喜梅[14]提出了协同环境下基于 RBAC 模型的访问控制策略,该策略针对共享资源访问控制策略的授权方式复杂、授权粒度不细致的问题,提出引入角色机制,并将角色访问并发控制策略应用到系统中。然而,该策略需要在协同开始之前就进行权限的预分配,其动态性较差。蒋永等人[15]提出了用户权限的灵活管理策略及动态分配技术,当企业组织结构、岗位人员动态调整时能够动态调整人员权限。该策略虽然提供了动态性,却仍然无法满足复制式架构的实时系统中对实体级的权限动态分配要求。Feng 等人[16]提出采用 Lock 机制结合 AST 技术实现协同编辑领域中用户编辑的语意维持,其通过加锁方法保证部分文档不被其他用户修改,从而实现该部分文档的修改权限控制。该策略虽然能够提供复制式架构下的访问控制,但由于其功能简单(仅能提供修改权限管理),且仅适用于所控实体的地址空间连续的环境中,无法适应 CAD 环境下同组实体地址离散的特点。此外,加锁机制也降低了实时协同系统的并发度。

笔者所在工作组曾提出支持团队分工的实时协同机制。该机制将文档按照不同团队的编辑请求划分为互不相交的任务区域,并通过修正本地操作执行广播过程及远程操作获取执行过程来支持各子文档的一致性。该策略较好地满足了不同组成员之间分工协作的需求,但该策略所处理的文档模型是线性文档,其基于文档分割实现分工协作的方式也同样不能适用于 CAD 环境中同组实体地址离散的特点。

本文针对 CAD 领域的文档模型特点,提出支持权限分工的层次结构文档模型,通过将权限信息附加到不同粒度的对象上,实现对层次文档各层次对象的权限控制。此外,本文提出在权限操作中附加时间戳信息,并采用地址回溯技术保证权限操作在各分布式站点以不中断用户操作的前提下得到正确执行,因此不会降低实时协同系统的并发度。

## 2　层次结构文档模型

实时协同领域中的操作转换及地址空间转换算法都是将文档影射成线性结构[4~7]。这种影射模式不符合设计领域的文档模型特点,且容易导致算法的复杂度增加(因为用户的 insert/delete 操作会导致从其作用点之后所有文档元素的位置都发生变化)。为此,根据 CAD 文档模型的特点,将其按层次抽象成层次结构,如图 1 所示。每一个节点由若干个元素组成,每个元素上按照时间戳向量[4,5]顺序附加一个线性的操作历史序列。第一层上的节点元素按照其在第一层次中线性表的位置来标志。第 $i$ 层的节点元素的标志方式采用$\langle P_1, P_2, \cdots, P_i\rangle$$(1 \leqslant i \leqslant 5)$的形式。其中:$P_j(1 \leqslant j \leqslant i-1)$表示该节点元素在第 $j$ 层上的祖先节点元素在本节点上的顺序号;$P_i$ 表示该节点元素在本节点上的顺序号,即对应图 1 所示结构;$P_1$ 表示该节点所在的文档序号;$P_2$ 表示该节点所在的图层序号;$P_3$ 表示该节点所在的图块序号;$P_4$ 表示该节点所在的实体序号;$P_5$ 表示该节点所在的顶点序号。例如,图 1 中节点元素 $V_{2,1,1,2,4}$ 表示该顶点是第 2 个文档、第 1 个图层、第 1 个图块中第 2 个实体的第 4 个顶点信息。

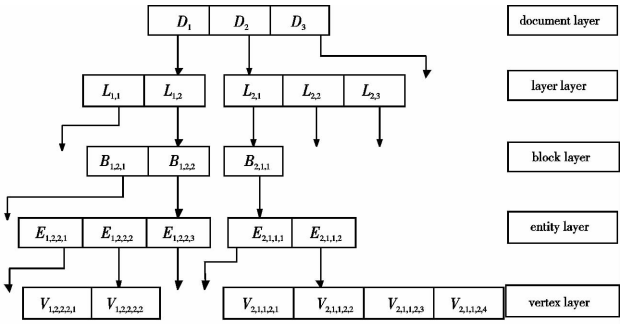


**图1　层次结构文档模型**

需要说明的是,节点元素的编号是与上下文有关的。不同的文档状态中,编号互不相同。例如,如果 $E_{2,1,1,1}$ 被执行 Delete 操作,那么 $E_{2,1,1,2}$ 的编号变化为 $E_{2,1,1,1}$;$E_{2,1,1,2}$ 的子节点 $V_{2,1,1,2,1}$、$V_{2,1,1,2,2}$、$V_{2,1,1,2,3}$、$V_{2,1,1,2,4}$ 则变化为 $V_{2,1,1,1,1}$、$V_{2,1,1,1,2}$、$V_{2,1,1,1,3}$、$V_{2}$,1,1,1,4。

层次结构文档中节点的定义请见定义 1,层次文档的定义请见定义 2。

**定义 1**　节点(node)。在层次结构的文档中(图 1),节点 $N$ 是 $n$ 个元素的有序列表$\{E_1, E_2, \cdots, E_n\}$,表中每个元素 $E=\langle$level, childPtr, entityID, historyPtr, mark$\rangle$:

a) level。节点 $N$ 在文档中的层次编号,level 取值 1、2、3、4 和 5,分别表示文档、图层、图块、实体和顶点。

b) childPtr。指向节点 $N$ 中的 $E$ 元素位于第 level + 1 层上的子节点的线性表的指针。

c) entityID。元素 $E$ 在当前 session 中的唯一标志符。

d) historyPtr。指向以该节点为目标对象的操作序列。

e) mark。用来表明当前元素是否有效,可取值 effective、ineffective。

**定义 2**　层次结构文档模型(layer-doc)。层次结构文档模型是 $n$ 个节点的有限集,它或者是一个空集($n=0$),或者由一个根节点及 $k$ 棵互不相交的子层次文档树组成。用 layer-doc 表示 document 层节点的唯一标志符,用 layer-doc$\langle P_1, P_2, \cdots, P_i\rangle$ 作为第 $i+1$ 层节点的唯一标志$(1 \leqslant i \leqslant 4)$,其中 $P_j(1 \leqslant j \leqslant i)$ 表示该节点位于第 $j$ 层上祖先在节点线性表中的顺序。例如,$\text{block}_{1,2,1}$、$\text{block}_{1,2,2}$ 所在的节点的标志为 layer-doc$\langle 1,2\rangle$。

关于层次结构文档模型下共享文档的一致性维护模型及算法,本研究组已做了相关的研究工作[17]。本文的重点不在于共享文档的一致性维护工作上,而在于共享文档中权限分工的设置及权限操作的意愿维持上。

## 3　支持权限分工的层次结构文档模型

文档不同部分需要根据实际需求设置不同的更新模式,即

可见/不可见、可写/只读。对于已经得到了需求确认的部分，由于其在后续操作中允许被修改，其模式被设置成只读模式（即不允许编辑人员再对本部分文档作修改）；另外，不同团队之间的产品只允许互相参照，而不允许相互修改，此时也需要将其他用户的产品设置为只读模式。此外，如果有的设计产品是保密的，不希望被后面参与的用户看到，该部分产品应该被设置为不可见。因此，要设置两种权限模式，即只读/可写、可见/不可见。

首先在每个节点元素上添加 visible 和 updateable 两个字段，分别用来表示当前的节点元素是否可见，是否可修改（包括可删除）。修改之后的文档节点元素结构如图 2 所示。

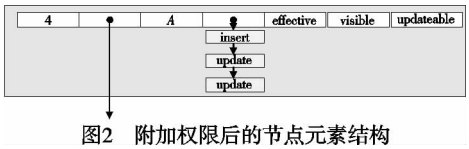


图2 附加权限后的节点元素结构

对于通过 insert 操作新产生的实体节点元素，根据具体的编辑要求有三种属性设置方式：

a）所有站点上该实体节点元素的 visible 和 updateable 属性都被设置为 true，表示该实体在所有的站点都是可见的，并且都是可以被修改的。

b）只有 insert 操作的 initiator 站点中该实体节点元素的 visible 和 updateable 属性被设置为 true，其他站点上 visible 属性被设置为 true，而 updateable 属性被设置为 false，表示该节点元素在所有站点上都是可见的，但只有产生该实体的站点拥有编辑权限。

c）所有站点的 visible 属性设置为 true，但只有与 insert 操作的 initiator 站点同组（team）的用户的 updateable 属性被设置为 true，其他工作小组用户的 updateable 属性均被设置为 false，表示该实体能够被同组内的所有用户所修改，其他小组用户仅拥有对该实体的查看权限。

## 4 用户角色设置

无论采用何种默认权限分配方式，分配好的权限均可以通过权限设置语句进行重分配或设置。为了支持权限设置，首先将项目中不同的用户按照职责进行分层定义，如图 3 所示。

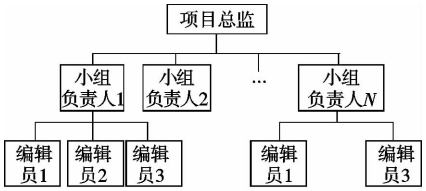


图3 用户角色层次表

一个项目中包含一个项目总监及若干个小组负责人，每个小组内又有若干个编辑人员。在用户角色层次表中规定：

a）上层的用户可以对下层的用户执行权限分配操作，如小组负责人可以控制本组内所有用户对所有对象的权限。

b）对管理同层用户的权限，提出修改的用户必须首先向上层用户提交申请，由上层用户负责对相关用户发出权限修改操作。例如，小组 1 中的参与者 1 希望 object1 不被小组 2 中的成员所修改，则参与者 1 首先向小组负责人 1 提出申请，由小组负责人 1 将此申请提交给项目总监，然后由项目总监通知小组负责人 2 中的所有成员执行权限设置操作。

## 5 权限及权限操作格式

根据前面的分析，定义了两种类型的权限：visible 和 updateable。这两个参数均可取值为 true 或 false，所表示的含义如下：

visible = true 为可见，visible = false 为不可见（这里要注意 visible 字段与 effective 字段的区别。Visible 字段只是表明了所参照的实体对象在用户界面上的可见性，而 effective 字段则表达了实体在实际文档空间中的有效性）。

updateable = true 为可修改；updateable = false 为不可修改，表示该实体对象只可读，而不能修改。

权限操作分为两种：权限申请语句及权限执行语句。权限申请语句用来由下层角色用户向上层角色用户提出权限修改的申请；权限执行语句用来由上层角色用户向下层角色用户发送执行权限修改的操作。权限操作的两种类型格式如下：

a）权限申请语句。apply（[权限]，[对象]，[角色]，initiator，*SV*），表示在 *SV* 状态下 initiator 站点申请将[对象]上的[权限]加载到[角色]上。其中：[权限]部分采用诸如 visible = true，updateble = false 的格式；[对象]采用〈$p_1$〉的地址来描述；[角色]采用〈$siteId_1$，$siteId_2$，…〉的格式来描述。例如，apply（updateable = false，〈1，1〉，〈1，2〉，3）表示站点 3 申请将第 1、2 个站点上第 1 个文档中的第 2 个图层上的实体设置为只读属性。

b）权限修改语句。grant（[权限]，[对象]，[角色]，initiator，*SV*），表示在站点[角色]上执行[对象]的[权限]分配操作。[权限]，[对象]，[角色]的含义与 apply 操作的一致。例如，grant（visible = false，〈1，2〉，〈3〉，2）表示在站点 3 上执行站点 2 发出的设置文档 1 的第 2 个图层不可见的操作。

权限的[对象]除了可以用站点 ID 值来表达外，还可以用以下几个带有整体含义的固定词汇来表达：

OTHERS：表示除了本地站点以外的所有其他站点。

TEAM#：表示用小组#中的所有成员（包括小组负责人和小组成员），也可以用 TEAM1，TEAM2，…等 TEAM 序列列表来表达多个小组的成员。

## 6 权限更新操作意愿维持

权限更新操作 grant（[权限]，[对象]，[角色]，initiator，*SV*）需要在相应的上下文环境下才能执行，即 grant 操作也是有状态的。如果有与 grant 操作并发的文档修改操作（insert，delete 操作），grant 操作在执行了与之并发操作的文档状态下执行，其[对象]参数就不能够得到正确的解释。因此，grant 操作的执行也必须首先经过文档的 retrace 过程，将文档状态回溯到 grant 操作产生时的状态，然后再在回溯后的状态下执行相应的权限更新操作。

### 6.1 Layer-AST 算法

在层次结构的文档回溯过程中，不需要比较文档中全部实体的地址空间，只需自顶向下回溯操作 *O* 所在节点的祖先节点及 *O* 所在的节点 NODE 即可。回溯过程的目标是将层次文

档中第 1 到 level(NODE) - 1 层中的 NODE 的祖先节点回溯到当前操作的时间戳向量 $SV$,然后通过比较 $SV$ 及节点操作的时间戳从而确定这些节点在时间戳 $SV$ 下是否有效。每一层节点中只需要回溯其中一个节点线性表的状态,因此该算法比 AST 算法提高了执行效率。回溯过程隐藏了所有与 $O$ 并发的 insert 和 delete 操作的执行效果(update 及 grant 操作不包含在内,因为 update 及 grant 操作不会改变线性表中实体元素的绝对位置,也就不会改变元素的唯一标志符),仅仅保留因果前序[4,7]与当前操作 $O$ 的操作的执行效果。假设节点 $N$ 的节点 $E$ 上的操作历史列表中的插入和删除操作所对应的时间戳分别为 $SV_{ins}$ 和 $SV_{del}$,该算法根据四种情况判断节点 $N$ 在时间戳 $SV_o$ 下是否有效[5,6,16]:

a) 如果 $SV_{ins} \geqslant SV_o$,则节点 $N$ 被设置为无效。

b) 如果 $SV_{ins} \leqslant SV_o$ 且无针对节点 $N$ 的删除操作,则节点 $N$ 设置为有效。

c) 如果 $SV_{ins} \leqslant SV_o$ 且 $SV_{del} \geqslant SV_o$,则节点 $N$ 为有效。

d) 如果 $SV_{ins} \leqslant SV_o$ 且 $SV_{del} \leqslant SV_o$,则此节点 $N$ 为无效。

由于节点元素的删除操作肯定在插入操作之后才能进行,因此元素 $E$ 上的插入和删除操作肯定满足 $SV_{ins} \leqslant SV_{del}$,所以不会出现 $SV_{ins} \geqslant SV_o$ 且 $SV_{del} \leqslant SV_o$ 的情况。如果存在多个并发的删除操作同时作用于节点 $N$,只要其中一个满足 $SV_{del} \leqslant SV_o$,则节点 $N$ 就被置为无效。层次结构文档模型下的 retracing 过程 1 如下:

**过程 1**　retracing_layer-doc (layer-doc, level, $SV_o$, $O$) 将 layer-doc 的文档状态回溯到操作 $O$ 产生时的状态

```
//分层次检查文档中每个节点的可见性
//假设 O. pos = {p1 ,p2 ,…,pn-1 , pn}
1 if level = = length( O. pos)
2    return;
3 for every element Ei of layer-doc {
4    set Ei ineffective;
5    consider the insert Oins of Ei , if Oins is timestamped by SVOins and
SVOins ≤ SVo {
6    set Ei effective;
7 }
8 for each delete Odel of Ei {
9    if Odel is timestamped by SVOdel and SVOdel ≤ SVo {
10      set Ei ineffective; } }
11 从左至右计算 layer-doc 中第 p[level] 个有效元素,记为 E[p[level]];
12 layer-doc = E[p[level]]. childPtr;
13 retracing (layer-doc, level + 1, SVo, O ); }
```

## 6.2　基于 Layer-AST 的权限意愿维持算法

权限更新操作 grant ([权限],[对象],[角色], initiator, $SV$) 从上层站点发出后,到达本地站点时,由于并发的 insert/delete 操作存在,有可能导致 grant 操作的 $SV$ 滞后于本地站点的当前状态。因此,必须首先调用 retracing 过程,将文档回溯到 $SV$ 状态,在 $SV$ 状态下执行 grant 操作,执行完成后,再将文档状态回溯到本地站点当前的状态。过程 2 描述了 grant 操作在本地站点的执行过程。图 4 描述了在文档状态(a)下执行 Grant 命令的过程。

**过程 2**　execute( grant, layer-doc, $SV_o$), 在 $SV_o$ 状态下执行 grant 操作

```
1 if grant is not causally-ready {queue( grant) ;}
2 else {
3    retracing_layer-doc (layer-doc, 1, SVo, grant )
4    execute( Grant) ;
6    SVo[ grant. initiator] = SVo1[ grant. initiator] + 1;
7    grant. SV = SVo ;
8    retracing_layer-doc( tree-doc, 1, SVo, grant) ;}
```

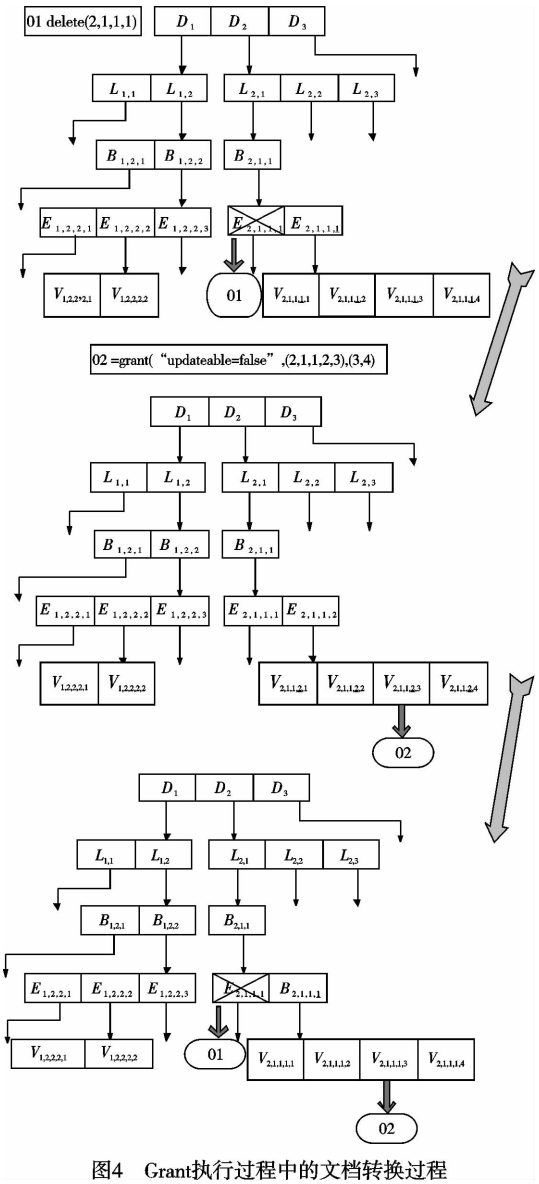


图4　Grant执行过程中的文档转换过程

## 6.3　效率分析

函数 retracing_layerdoc (layer-doc, level, $SV_o$, $O$) 回溯层次结构文档 layer-doc 的地址空间到第 level 层的操作 $O$ 的时间戳 $SV_o$ 所表示的文档状态下。每次回溯过程中不需要与层次文档中所有的节点元素进行比较,只需要对同一层次的其中一个节点进行比对即可。因此,该算法与 AST 算法相比提高了执行效率。

假设层次结构文档中节点线性表的平均长度为 $l$,即回溯过程每一层次上需要与 $l$ 个节点元素的时间戳向量进行比较,

以确定该节点元素的有效性。假设层次结构文档每个节点元素所对应的编辑操作数量为 $d$,则每个节点有效/无效标记的检查需要时间花费 $O(d)$,于是回溯过程的渐进效率可以表示为 $O(d \cdot l)$。因此,retracing 过程的总效率为 $O(l \cdot \text{level} \cdot d)$,level 值由 $O$ 所在的层次决定。

AST 中的 retracing 过程需要对 doc 中的最后一层节点的所有元素进行回溯。根据前面关于线性表平均长度的假设推断出层次树最低层节点元素的个数为 $h^{\text{level}-1}$,则 AST 算法的回溯执行效率为 $O(h^{\text{level}-1} \cdot d)$。显然,Layer-AST 的执行效率更高。

## 7 站点 ID 的设置及初始化讨论

为了保证本地站点的用户能够获知自己的其他工作人员所处的位置,以便能够准确地定义权限操作中的[对象]参数。在每个站点中都维持有一张用户角色层次表,该表的逻辑结构如图 5 所示。位于第一层次的是该项目的项目负责人运行实例的序号及 IP 地址和端口号;位于第二层次的则是各小组负责人运行实例的序号及 IP 地址和端口号;位于第三层次的是小组内成员的信息。

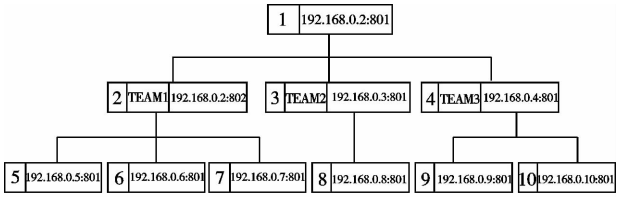


图5 用户角色层次逻辑结构

在该层次结构中,每个站点除了有一个用 1,2,…数字信息表达的全局顺序外,还有一个与角色有关的序号。例如 super1,super2,…表示项目总监 1、2 等;team1,team2,…表示小组负责人 1、2 等;des1, des2,…表示设计员 1、2 等。在协同工作初始化时,首先由项目总监登录到 session 管理服务器,由管理器为总监进程分配相应的站点 ID 值。在项目总监登录之后,各小组负责人及普通设计员再依次进行登录。低层用户在登录过程中要选择自己所在小组,并由服务管理器分配站点 ID 值和角色序号值,并构建用户角色层次树。在所有的用户完成登录后,管理器启动协同命令,并将该 session 中所生成的用户角色层次树广播给所有用户。之所以采用层次登录的方法,是为了避免角色层次树在构建过程中不断地进行调整操作,从而减少操作时间,提高处理效率。

## 8 结束语

实时协同系统中通常采用无限制协同模式,虽然其增加了系统的并发度,但各个参与成员拥有统一的权限标准,并能够对共享文档的任意对象执行任意编辑操作,导致权限分工的执行只能依靠外部协议(如口头协议)。如果有用户对外部协议不能严格执行,则会破坏整个工程的工作效率,缩小实时协同的适用范围。本文通过构建用户角色层次表维护不同设计人员之间的角色关系,定义支持权限分工的树型结构文档模型及权限操作格式,讨论了站点 ID 的设置及系统初始化过程,并提出通过改进的地址空间转换算法来支持权限操作的意愿维持。权限分工策略已被应用到笔者之前所开发的 Co-AutoCAD 实时协同原型中,得到了协同参与人员的认可和肯定。

进一步的工作包括将权限管理策略运用到支持团队分工的实时协同环境中,在保证高响应度及并发度的前提下,又同时支持不同角色设计人员的分工合作,以更好地满足协同设计环境中用户的协同需求。

**参考文献:**

[1] XIA S,SUN D,SUN Cheng-zheng,*et al*. Leveraging single-user applications for multi-user collaboration:the CoWord approach[C]//Proc of ACM Conference on Computer Supported Cooperative Work. New York:ACM,2004:162-171.

[2] 史美林, 向勇, 杨光信. 计算机支持的协同工作理论与应用[M]. 北京:电子工业出版社,2000.

[3] SARMA A, Van der HOEK,PALANTIR A. Coordinating distributed workspaces[C]//Proc of the 26th Annual International Computer Software and Applications Conference. 2002:1093-1097.

[4] ELLIS C A,GIBBS S J. Concurrency control in groupware systems[C]//Proc of ACM SIGMOD International Conference on Management of Data. New York:ACM,1989:399-407.

[5] SUN D,SUN Cheng-zheng. Operation context and context-based operational transformation[C]//Proc of the 20th Anniversary Conference on Computer Supported Cooperative Work. New York: ACM, 2006: 279-288.

[6] GU N,YANG J,ZHANG Q. Consistency maintenance based on the mark & retrace technique in groupware systems[C]//Proc of International ACM SIGGROUP Conference on Supporting Group Work. New York:ACM,2005:264-273.

[7] LI D, LI R. An admissibility-based operational transformation framework for collaborative editing systems [J]. Computer Supported Cooperative Work,2010,19(1):1-43.

[8] 范明虎,樊红,伍孝金. ASP. NET 中基于 RBAC 的通用权限管理系统[J]. 计算机工程,2010,36(1):143-146.

[9] 赵静,杨蕊,姜滦生. 基于数据对象的 RBAC 权限访问控制模型[J]. 计算机工程与设计,2010,31(15):3353-3355.

[10] 朱磊,周明辉,刘天成,等. 一种面向服务的权限管理模型[J]. 计算机学报,2005,28(4):677-685.

[11] 杨飞,袁建华. Hibernate 框架在权限管理系统中的应用[J]. 微电子学与计算机,2007(2):206-208.

[12] 林伟炬,刘列根,张宇. 一个通用的权限管理模型的设计方案[J]. 微计算机信息,2009(15):1-3.

[13] 方萃浩, 叶修梓, 彭维,等. 协同环境下 CAD 模型的多层次动态安全访问控制[J]. 软件学报,2007,18(9):2295-2305.

[14] 付喜梅. 协同环境中基于 RBAC 模型的访问控制策略[J]. 计算机工程,2009,35(11):140-142.

[15] 蒋永,蒋玉明,彭思达. 基于工作流用户权限管理模型的研究与设计[J]. 计算机技术与发展,2009,19(1):161-164.

[16] FENG Ya-hui,GAN Li-ping,GU Ning,*et al*. Locking intention preservation based on address space transformation technique[C]//Proc of the 3rd International Conference on Pervasive Computing and Applications. 2008:497-502.

[17] GAN Li-ping,LU Tun. Achieving transparent & real-time collaboration in Co-AutoCAD application[J]. Journal of Universal Computer Science,2011,17(14):1887-1912.