一种开放式超媒体系统版本模型*

李光亚 周学海 赵振西

(中国科学技术大学计算机系 合肥 230027)

摘要 本文提出了一种开放式超媒体系统的版本控制模型,它支持版本集、允许可选构件的开发和管理、维护文档的发展历史并且支持协同工作;提出了上下文结点和版本相关表的概念,用以有效地解决结构信息版本和内容信息版本之间的不一致性问题.

关键词 版本控制,开放式超媒体系统,版本相关表,上下文结点.

中图法分类号 TP311.52

自从Halasz提出将版本控制作为超媒体系统的一个研究方向以来[1],超媒体系统的版本控制已引起越来越多的重视.超媒体系统已经被广泛用于各种开发性的任务[2],在这些任务中,用户需要版本支持来进行诸如写作和思想组织的活动;同样,超媒体系统也被应用于软件构件管理和协同工作,在这些任务中则需要更加广泛的版本控制机制.

当前对于超媒体系统版本控制的研究绝大多数限于封闭的超媒体系统和特定的应用领域,开放性已成为超媒体领域最重要的研究方向.[3]而实现开放性最主要的基础[3]就是超媒体的结构信息和内容信息的分离,结构信息通常是由超媒体系统来处理;而内容信息的编辑则是由外部应用系统来完成的,而这些应用系统一般都有各自的版本管理机制.[4]显然版本控制机制在开放式超媒体系统中要比在那些独立封闭的超媒体系统中复杂得多.

基于上述情况,本文首先详细分析了应用对版本控制的各方面特性的需求,进而论证设计了满足这些需求的有关版本控制机制,建立了功能完备的开放式超媒体系统版本模型.

1 版本机制的需求

我们先简要地列出在文献[1,2,5]中所提出的对超媒体系统中的版本控制机制6个方面的需求,这些也正是设计模型所要考虑的主要问题.

R_1:所有超媒体对象(结点、链和锚)的版本化,而不仅仅是文档内容的版本化.

R_2:超媒体结构的版本化.这意味着不仅链和锚的版本化,对于超媒体网络的一部分或全部都应该可版本化,它允许用户返回到超媒体文档发展历史中某时刻的一致性状态.

R_3:版本的选择.当用户要存取一个版本化的超媒体对象时,就牵涉到具体版本的选择问题,超媒体的版本控制机制应该尽量减少用户的认知负载,并提供一定的选择灵活性.

R_4:版本的产生.何时产生新的版本对超媒体系统是非常重要的,版本产生过多将影响系统的效率和增加用户认知负载,版本产生过少则不能完整地记录文档的发展历史.

R_5:支持版本分支的产生.某些临时性的修改或并发工作会导致版本分支的产生.

R_6:对于多用户协同工作的支持.对于多个合作者同时对超媒体网络信息进行读写,超媒体系统应该提供一种有效的版本控制机制来协调他们的工作.

除了上述要求之外,对于开放式超媒体系统,版本控制模型应该提供以下额外的机制.

R_7:版本的不变性.版本代表了对象发展过程中的一个完整状态,是否允许对于一个版本化的对象进行修改,如进行注释或增加新的属性?

R_8:对于存取超媒体内容信息的第三方应用应进行最小限度的修改和扩充,以便能充分利用超媒体系统所提供的版本控制机制.

---

* 作者**李光亚**,1973年生,博士生,主要研究领域为超媒体系统,数据库系统.**周学海**,1963年生,博士生,讲师,主要研究领域为多媒体数据库,面向对象程序设计.**赵振西**,1937年生,教授,主要研究领域为软件环境,AI,多媒体信息处理.

本文通讯联系人:李光亚,合肥 230027,中国科学技术大学计算机系

本文1997-01-08收到原稿,1997-03-17收到修改稿

© 中国科学院软件研究所 http://www.jos.org.cn

R_9:超媒体内容版本和结构版本之间的一致性.既然内容信息和结构信息各自独立发展而产生各自的版本,应该提供一种对用户尽可能透明的机制来保证二者的同步关系.

R_{10}:超媒体系统所提供的版本数据模型应该具有通用性,不能偏向某一类特定的应用.

2 数据模型

对于超媒体文档的定义是基于3种简单抽象的概念:链、锚和结点.结点是具体或抽象的信息片断;锚是结点的外部接口,它具体定位了结点内容中的某一部分,它的值由外部应用对其进行解释并加以显示;链则通过锚来关联各个结点.它们的具体分类如图1所示.

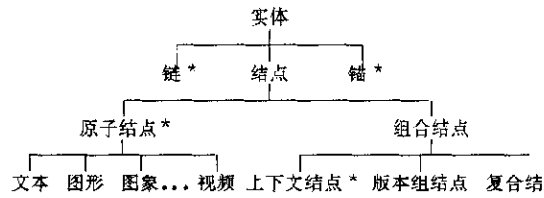


图1 数据模型

2.1 基本概念

原子结点的内部结构依赖于具体的第二方应用,对于超媒体系统来说是透明的.组合结点是由其它的组合结点、链和锚组成,它可以分为3类:上下文结点、版本组结点和复合结点.链由一个源端点和一系列目标端点组成,有多个目标端点时链可用来定义一对多的连接.链的端点由结点描述符(N_i,V_i)和锚描述符(A_j,V_j)组成,版本号用来指定特定版本的结点和锚.链是有向的并且存储在超媒体系统中的链库中.为了能够描述某一类结点、链和锚的变化历史,我们引入了版本组结点的概念,这种组合结点的组成成分都是某一相同实体的不同版本,版本组结点的内部结构为树型,树中的边即链代表了不同版本之间的导出关系,例如,若版本树中有一条边$((N_i,V_1A_j,V_j)(N_i,V_2A_k,V_k))$,则表明$N_i$结点的$V_2$版本是从$V_1$版本变化而来,此链的锚用来说明$V_1$的哪一部分通过变化产生了$V_2$的相应部分,这满足了$R_5$的需求.对于复合结点的概念我们沿用了Dexter参考模型[6]中的定义,它是由别的原子结点、复合结点和链所组成的,用它可以描述超媒体网络中需要一致性变化的子网结构.图1中所有加*号的实体都是可版本化的,而超媒体系统是由链、锚、原子结点和复合结点所组成的,这满足了R_1和R_2的需求.所有这些可版本化实体是通过两步定位机制来存取的:首先在版本组结点中找到相应的实体对象(如N_i),然后再根据版本号(如V_j)在版本组结点中找到该实体的特定版本.

2.2 上下文结点(Context node)和版本相关表VAT(version association table)

由于在开放式超媒体系统中结构信息和内容信息分离,可以分别对它们进行编辑而产生各自的版本,这种独立的发展会导致结构和内容信息的不一致问题,即当对一个结点的内容进行浏览时,选取与该内容版本相对应的结构信息的哪一个版本.为此我们引入了上下文结点和版本相关表的概念.

上下文结点是一类特殊的组合结点,它所包含的是同某一结点(原子或复合)版本相关联的所有链版本,可见它所表示的是特定结点版本在整个超媒体网络中的上下文概念.每个链在任一上下文结点中最多可被表示为一个链版本,但同一链版本可以出现于多个上下文结点中,图2给出了一个超媒体子网以及上下文结点的例子.

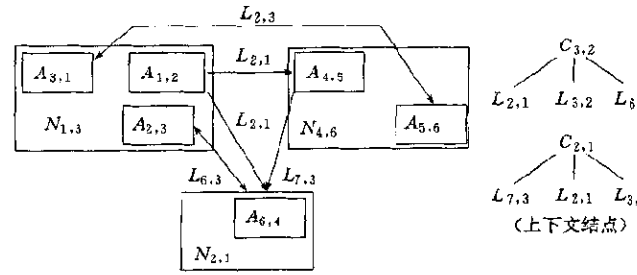


图2 超媒体子网及上下文结点示例

由图2可见,上下文结点的主要目的是,定义在对某一结点版本内容进行修改时可能涉及到的所有结构信息的版本.上下文结点是可版本化的,因为在对结点内容进行编辑的同时,可能会修改或删除所含的结构信息,如用户在编辑$N_{1,3}$而产生新的结点版本$N_{1,4}$时,可能将$A_{2,3}$的位置发生改变,这时超媒体系统应该能够将$C_{3,2}$中链$L_{6,3}$的版本进行更新而产生新的链版本$L_{6,4}$(包含新的锚版本$A_{2,4}$),并让它包含在新的上下文结点版本$C_{3,3}$中.如何具体地将锚$A_{2,3}$更新为$A_{2,4}$,是所有开放式超媒体系统都要解决的问题.它是由锚定位机制所决定的,需要由超媒体系统借助于特定外部应用的知识来完成.[7]应用本模型应该更加容易实现,因为从$N_{1,3}$到$N_{1,4}$的变化都已被记录于相应的版本组结点中.

从以上讨论可以看出,结点版本(原子或复合)和上下文结点的版本是一一对应的,如图2中$N_{1,3}$和$C_{3,2}$,$N_{4,6}$和$C_{2,1}$.为了记录这种映射关系,我们引入了版本相关表的概念,它将结点的版本映射到与之相对应的上下文结点的版

© 中国科学院软件研究所 http://www.jos.org.cn

本，此表由系统自动建立和管理. 与图 2 相对应的版本相关表如表 1 所示.

表1　版本相关表

结点版本	上下文结点版本
$N_{1,3}$	$C_{3,2}$
$N_{4,6}$	$C_{2,1}$
.....	.....

系统提供一种检索功能，当给定一结点版本时返回与之相关联的上下文结点版本或者反之. 由此可见，版本相关表和上下文结点的概念部分满足了 R_7 的需求. 为了满足 R_9 中的透明性要求，系统自动为每个活跃的用户建立一个活跃的上下文集合，它包含了所有与当前正被打开的结点版本相对应的上下文结点版本. 每当一个用户浏览到一个新的结点之后，与此结点相对应的上下文结点的版本通过检索版本相关表而被动态得加入到该用户的活跃上下文集合中去. 同样，若用户关闭了一个已经打开的结点，那么相应的上下文结点版本由系统从该用户的活跃上下文集合中除去.

3　版本控制机制

3.1　用户的浏览行为

当用户打开一个结点版本，可对此结点所包含的结构信息进行浏览时，分 3 种情况.

(1) 当结点 iN_i 是刚生成的(V_i-1)，还没有包含任何超媒体网络信息，则处理算法为：系统为 $N_i,1$ 建立一个空的上下文结点版本 $C_i,1$，将($N_i,1$ $C_i,1$)加入版本相关表，$C_i,1$ 加入该用户的活跃上下文集合中去；

(2) 当结点(N_i,V_i)是以前生成的版本，并且与此版本所对应的超媒体结构信息已经建立过，则处理算法为：

(1) 系统在版本相关表中查找 N_i,V_i 这一项，找到相应的上下文结点版本 C_j,V_j；

(2) 根据 C_j,V_j 中所有的链版本查找链库；

(3) 从链库中找到 C_j,V_j 中所包含的 N_i,V_i 的锚，根据其版本对它们进行解释并加以展示；

(4) 将 C_j,V_j 加入该用户的活跃上下文集合中去；

(5) IF 用户选择了某个锚进行导航；THEN

(5.1) 根据此锚的版本查找活跃上下文集合中的链；

(5.2) 找到以该锚为源端点的链的所有目标结点及链目标端点的锚；

(5.3) 打开(5.2)中所有的目标结点(递归调用)；

END

(3) 当此结点(N_i,V_i)是刚生成的版本，与此版本相对应的超媒体结构信息还没有建立(VAT 中没有与 N_i,V_i 对应的条目)，但是此结点的前趋版本已经建立了超媒体网络信息，则处理算法为：

(1) 系统从包含 N_i 结点的版本组结点中找到该结点的前趋版本 N_i,V_{i-1}；

(2) 从版本相关表中查找与 N_i,V_{i-1} 相对应的上下文结点 C_j,V_{j-1}；

(3) 从 C_j,V_{j-1} 中查找结点 N_i,V_{i-1} 中所包含的所有锚 A_k,V_k；

(4) 根据版本组结点中从 N_i,V_{i-1} 到 N_i,V_i 的变化过程对(3)中改变的锚产生新的版本 A_k,V_{k+1} *

(5) 将(4)中产生的新的锚版本记录到版本组结点中去；

(6) IF C_j,V_{j-1} 中有链 L_m,V_n 的源端点或目标端点发生改动，THEN

(6.1) 产生新的上下文结点 C_j,V_j；

(6.2) 产生新的链版本 L_m,V_{m+1}，将链的源端点或目标端点改为(4)中的新版本；

(6.3) 将 C_j,V_j 中 L_m,V_n 替换为 L_m,V_{n+1}；

END

(7) 将(N_i,V_i C_j,V_j)或(N_i,V_i C_j,V_{j-1})这一项加入版本相关表中；

(8) 将 C_j,V_j 或 C_j,V_{j-1} 加入该用户的活跃上下文集合中.

3.2　对并发行为的支持

除了记录超媒体文档的发展历史之外，版本控制的另一主要目的是减少并发编辑操作的冲突. 在多个用户同时对一超媒体文档进行编辑时，版本控制可以在一定程度上支持它们之间的协同工作. 由于超媒体结点内容的编辑是可以

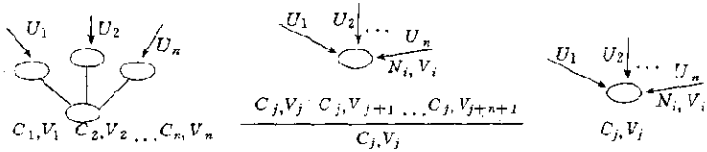
---

* (4)中建立新的版本是要根据版本组结点中从前趋结点到当前结点的改变而确定的，如位置的改变等. 这和如何实现锚的定位机制有关，不同的系统可能有不同的实现方式.[7]还有可能是新结点删除了其前趋结点中某个锚所对应的部分；若包含此锚的链不再关联结点 N_i,V_i，则此链版本从 C_j,V_j 中删去；否则产生新的链版本，并替换 C_j,V_j 中旧的链版本，这在(6)中没有给出. 以上(7)和(8)中若与结点 N_i,V_{i-1} 相对应的结构信息没有发生变化，则不产生新的上下文结点版本 C_i,V_i. 由此可见，本模型对于用户的浏览行为并没有增加认知负载，同时对于第三方应用没有增加新的版本功能需求，因此这满足了 R_8 的需求.

© 中国科学院软件研究所 http://www.jos.org.cn

独立于超媒体之外而进行的，这里我们只讨论对超媒体结构信息的编辑.

当多个用户对同一超媒体文档浏览时，系统不用加以控制，甚至不同用户可以同时浏览相同结点的不同版本，因为它们对应着各自的上下文结点版本.而当有多个用户同时编辑超媒体文档时，系统必须考虑到他们之间的合作方式.本模型支持图3所示的3种合作模式.[5]



(a) 个人模式 (b) 松耦合模式 (c) 紧耦合模式

图3 三种合作模式

个人模式允许不同用户工作在不同的结点中，他们分别对不同结点版本产生相应的上下文版本，系统提供一定的锁机制来控制多个用户对同一结点的存取.而松耦合模式和紧耦合模式允许不同用户对同一结点进行编辑，但松耦合方式中各自的编辑仍保持独立性，系统可以通过提供一定的事件通知机制让用户相互了解各自的工作进展以减少写冲突：每个用户的编辑行为通知给服务进程，而服务进程再将这些事件传递给其余用户方应用的事件处理器.为了减少用户应用响应事件的频率，每个用户U_i可用事件预约的方式向服务进程登记其感兴趣的事件(如U_i只想了解U_j与U_k的行为).松耦合模式中U_1~U_n对N_i分别产生了不同的上下文结点版本分支$C_j,V_jC_j,V_{j+1},\ldots C_j,V_{j+n-1}$，于是这些用户仍可以并发工作而不至于互相等待，当编辑过程完成之后，系统对$C_j,V_jC_j,V_{j+1}\ldots C_j,V_{j+n-1}$这$n$个上下文结点进行合并[8]，产生单一的上下文结点$C_j,V_j$作为与$N_i,V_i$结点相对应的最终上下文结点并加入到版本相关表中.紧耦合模式即用户之间遵循WYSIWIS的原则，可以通过只产生一个上下文结点版本来实现，从U_1到U_n所有用户对N_i结构信息的修改均体现在C_j,V_j中，即任何对N_i,V_i所做的修改均通知给服务进程，再由服务进程广播给每个用户方应用的事件处理器，事件处理器根据消息的内容对本应用窗口中N_i,V_i的内容做相应的改动，这就保持了各个用户所见的一致性.这3种模式部分满足了R_6的需求，同时也满足了R_5的需求，因为对不同的合作模式，系统产生了不同数量的实体版本，既完整地记录了文档的发展历史，又去除了某些不必要的中间版本.

3.3 版本的不变性和版本的选择

在合作开发超媒体文档的过程中允许用户对他们所生成的版本进行注释[1,2]是非常必要的.同时在开放式超媒体系统中，由于外部工具的引入，可能会需要对已有的结点增加新的属性以便使用这些工具，这也是非常必要的.由于内容信息和结构信息分离存储，用户可以很容易地对一结点版本内容进行注释，而且对结点的注释和对结点内容的编辑可以同时进行.同样，增加属性的工作也是可以脱离外部应用而在超媒体系统内完成的，满足了R_7的需求.

在基于本模型的系统中，一旦结点版本被确定，其上下文结点版本由版本相关表唯一确定.此外，允许用户通过显式地指出版本号，对任一可版本化的实体进行访问.如用户在访问N_i,V_i时，想了解其前趋结点的内容，用户只需指出版本号V_{i-1}，系统将活跃上下文集合中对应于N_i,V_i的上下文结点版本替换为与N_i,V_{i-1}相对应的上下文结点，从而可以满足R_3的需求.

4 结 论

本模型通过上下文结点和版本相关表等各种机制的提出，有效地解决了内容信息和结构信息版本之间的一致性问题.因此对于内容信息的管理可以利用已有的版本管理系统，又可以通过超媒体系统所提供的版本机制实现与应用系统的有效集成.此外，本模型是建立在简单抽象的概念：结点、链和锚之上，可以应用到众多的应用系统中去，满足了R_{10}的要求.

参考文献

1 Halasz F G. Reflections on notecards: seven issues for the next generation of hypermedia systems. Communications of the ACM, July 1988, 37(7): 836~855

2 Osterbye K. Structural and cognitive problems in providing version control for hypertext. In: Proceedings of the European Conference on Hypertext'92. Milan, Italy, Dec. 1992

3 Leggett J ed. Position papers of 2nd workshop on open hypermedia systems. In: Proceedings of ACM Hypertext'96. Washington, Mar. 1996

© 中国科学院软件研究所 http://www.jos.org.cn

4 Tichy. RCS——a system for version control. Software Practice and Experience, Jul. 1985,15(7):637～654
5 Will U, Leggett J. Concurrency control in collaborative hypertext systems. In: Proceedings of ACM Hypertext'93. Seattle, Nov. 1993. 14～24
6 Halasz F G, Schwartz M. The dexter hypertext model. Communications of the ACM, 1994,37(2):30～39
7 Vanzyl A J. Open hypermedia systems: comparisons and suggestions for implementation strategies. In: Proceedings of the European Conference on Hypertext'94. Edinburgh, UK, Sep. 1994. 11～15
8 Haake A. On merging hypertext networks. In: Proceedings of the 4th European Conference on Computer Supported Cooperative Work'95. Stockholm, Sweden, Sep. 1995

A Model of Version Control in Open Hypermedia Systems

LI Guang-ya ZHOU Xue-hai ZHAO Zhen-xi

(*Department of Computer Science University of Science and Technology of China Hefei* 230027)

Abstract This paper presents a version control model of open hypermedia systems among other features, which supports version sets, permits exploring and managing alternate configurations, maintains document histories and supports cooperative work. The concepts of context node and version association table are proposed to solve the problem of inconsistency between structure version and content version effectively.

Key words Version control, open hypermedia system, version association table, context node.

Class number TP311.52

© 中国科学院软件研究所 http://www.jos.org.cn