版本管理系统的设计方法

徐向阳

（华东师范大学计算机系）

摘　要

Version Management (VM) provides mechanisms for storing, accessing and inquirying each version, reducing space costs and recording modification information. An integrated VM design includes hierarchical designs of Configuration, module and text, VM tools should be rich in understanding of version semantics and relations to each other. A guideline for hierarchical design of VM is presented in this paper.

一、版本管理工具

计算机软件系统在其整个生命周期内需要经常修改、校正、扩充和改进，使系统产生多个版本，统称为系统簇。随着系统簇的不断膨胀，必须有效地管理和维护系统簇。软件配置管理SCM (Software Configuration Management) 是用来管理和维护系统簇的软件，其功能为：对源代码修改的控制和跟踪，配置生成，出错的跟踪与报表，以及产品的版本标志和管理。版本管理VM (version Management) 是支持SCM的一个工具，它应提供机制存贮、更新和查询软件的各个版本，减小软件版本所占存储空间，控制版本访问权限，以及记录版本形成时状态，如版本修改的原因、内容和修改者等。

先介绍两个概念：顺序版本和平行版本。

·顺序版本：因错误校正等原因，对版本进行一系列局部修改，按时间先后顺序生成的一连串文本。

·平行版本：由于软件模块的不同实现方式或功能扩展而形成一个以上的版本。

1. 源代码控制系统

最早的VM工具是在UNIX下运行的源代码控制系统SCCS（Source Code Control System, 1975），它是一个文本控制系统。为减小存贮开销，它只存放软件的初始版本和版本变更，为便于管理版本，限定版本访问权限和对版本附加标志和信息，并由程序带入目标文件；还可查询用户加入的注释信息和形成软件的修改报表。

2. 修改控制系统

AT & T公司于1982年发行了在UNIX下运行的修改控制系统 RCS (Revision Control System)。用户可指定文件名称和给出版本编号，以管

参考文献

1. Software Engineering Environment, Proceedings of the International Workshop on Software Engineering Environment. China Academic Publishers.
2. Adele Goldberg and David Robson. Smalltalk-80, The Language and its Implementation. Addison-Wesley, Reading, Massachusetts, 1983.
3. M. A. Jackson. "System Development", Englewood Cliffs, NJ. Prentice-Hall, 1983.
4. B. Stroustrup. "What is Object-Oriented Programming", IEEE Software, May 1988.
5. G. Booch. "Object-Oriented Development", IEEE Trans, Soft, Eng, Vol. SE-12, No. 2.
6. JAMES MARTIN. "SYSTEM DESIGN PROVABLY CORRECT CONSTRUCTS".

理多层平行版本；还用锁的办法防止当前版本被意外"取版本"操作所覆盖。

3． 变更控制系统

为提高目标码生成速度，变更控制系统CCS（Change Control System）允许互不干扰地同时开发同一产品的多个版本和计算目标模块间最小差异。

4． 依赖语义网的源代码控制系统

SCCS与RCS都限定顺序版本和平行版本，不便于多模块软件系统的描述。为克服此缺点，采用语义网描述版本间二元关系。

5． 编辑版本管理系统

为提高编辑效率，把版本控制与编辑结合起来。在VM/SP下运行的管理多版本的程序P-EDIT，允许同时开发多个版本。对顺序版本，计算并存贮版本间的差异；对平行版本，把变更部分代码段直接插入多版本程序并形成该段的控制信息，以供形成和显示版本。在修改控制编辑器EH（1987）中，版本正文以行为单位，并按树形结构存贮，这样新版本的形成是构造新子树，再与部分老子树合并而得，它提高了版本存取速度。

二、版本控制

研究版本管理，要考虑衡量版本控制机制的直观标准。

1． 版本标志

用于区分各版本，以便于人们的理解和计算机查找。

2． 版本等价性

对于可执行版本，其等价性是指两个版本可替换使用而不产生副作用。若两个版本（模块）的名称相同，参变元的名称、类型和顺序都相同，则称两个版本语法上等价。语法上等价的两个版本其功能不一定相同，因此，必须从语义上描述版本的等价性。

版本接口由前置条件集、后置条件集和后制动作集组成，其中，前置条件集描述版本执行前的条件，后置条件集描述版本执行后的条件，后制动作集描述版本执行后必须立即执行的动作。两个版本是等价的，当且仅当它们的前置条件集、后置条件集和后制动作集都相同。

3． 版本一致性

假若某版本要调用它版本，或由不同版本构成新版本，则要考虑版本之间的一致性。

复合版本：由系统中若干个版本构成。

语法一致：一个版本与它版本是语法一致的，当且仅当它们构成的复合版本无语法错误或类型校验错误。

语义一致：一个版本与它版本是语义一致的，当且仅当它们构成的复合版本无任何错误且功能符合规范。

4． 版本兼容性

版本等价性要求功能完全相同，但在实际上用某版本替换其它版本时，只要求部分功能匹配即可，即版本兼容性。

（1）严格兼容　版本V_2是V_1的严格兼容版本当且仅当满足下列条件：

1）V_1前置条件值域包含V_2前置条件值域；

2）V_2后置条件值域包含V_1后置条件值域；

3）后制动作集相等。

严格兼容版本具有可替换性。

（2）向上兼容　版本V_2是V_1的向上兼容版本当且仅当满足下列条件：

1）V_1前置条件值域包含V_2前置条件值域；

2）V_2后置条件值域包含V_1后置条件值域；

3）V_2后制动作包含V_1后制动作。

向上兼容版本保留且扩充原版本功能。

（3）实现兼容　严格兼容与向上兼容版本要求保留原版本全部功能，当只考虑保留部分功能时，则需放宽限制，为此引入实现兼容概念。

假定版本由接口说明与实现体两部分组成，实现体部分的前置条件集、后置条件集与后制动作集称为版本传播接口。

版本V_2与V_1严格实现兼容当且仅当两版本的传播接口相同。版本V_2是V_1的强实现兼容版本当且仅当V_1的传播接口是V_2传播接口的严格兼容版本。

等价版本必是严格实现兼容，但反之不然。严格兼容与严格实现兼容都是强实现兼容。

若不考虑用版本V_2替换V_1所产生副作用，则称版本V_2是V_1的弱实现兼容版本，因此可以说，任何版本都可被看成是其它版本的弱实现兼容版本。

（4）系统兼容　若版本V_1用版本V_2替换处处满足实现兼容性，则称V_2与V_1系统兼容。V_2与V_1是严格（强、弱）系统兼容，当且仅当在系统出现V_1的任何地方，V_2都是V_1的严格（强、弱）实现兼容版本。

根据实现兼容层次可划分系统兼容层次。各种兼容之间的层次关系如图1所示。

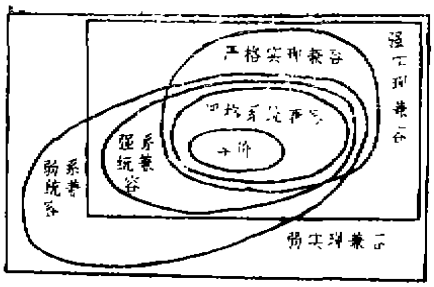

图1 兼容的层次

5. 版本控制机制

(1) 无版本控制 开发者只存放当前版本或某些顺序版本。程序员简单地把若干版本凑成一个复合版本，不考虑配置的规范；版本标志、等价性和兼容性由程序员决定；版本一致性通过编译、连接、运行和测试过程来判定。

(2) 基本版本控制 用版本标志区分和访问版本，它是用句号'.'分隔的自然数序列，如1.5。复合版本由成员名和版本标志序列组成，系统按列表或版本间隐含依赖关系形成复合版本，如{det 1.5, Jian 3.9.1.1}就由det 1.5和Jian 3.9.1.1所组成。版本等价性和兼容性由程序员确定，而一致性通过编译、连接、运行和测试过程来判定。

基本版本控制对象是模块或文件。

(3) 强类型版本控制 强类型版本控制对象是语法体，如过程、函数、数据结构等。等价指语法上的等价，自动验证版本的语法一致性，语义一致性在版本的生成与测试过程中确定。

(4) 语义版本控制 它达到语义上的等价，根据系统兼容的层次（严格、强、弱）选择实现兼容版本，并进行强类型校验，自动完成版本替换。生成复合版本时，自动验证语义一致性。用向上兼容概念区分平行版本和顺序版本。

在语义版本控制机制下，系统对版本等价性、一致性和兼容性的理解更接近现实。

三、系统设计方法

在操作系统之上建立版本控制缺乏一致性且对象易变；而在操作系统中集成版本控制和配置控制有利于分布式软件开发。在大型软件开发环境中，版本管理控制要考虑高层次系统配置，以减轻配置管理和用户负担。

版本管理系统由若干配置组成，配置由若干模块组成，而模块可有多个版本。配置层提供机制去控制组成配置的各模块和复合版本；模块层提供机制来管理一模块的多个版本；文本层实现有效地存贮多版本文本。

1. 配置层的设计

系统的模型定义了系统成员即模块、文档和程序员等之间的相互依赖关系。系统设有：配置表，它记录配置生成的历史信息；修改请求(MR)表，它记录提交的修改请求和采取的相应处理。配置层根据这两张表组成复合版本。

系统有两种方式支持程序员的修改：一是开放方式，即程序员的修改会导致系统的修改，如源程序修改后，系统自动编译、连接和装配；二是保护方式，即程序员的修改不影响其它程序工作，除非修改完毕并已被确认。被删除版本要保证不再被其它模块所调用。

生成复合版本时，系统按模块接口说明与系统兼容性要求，自动选择实现兼容版本，并进行版本一致性自动验证。

2. 模块层设计

模块由接口、文本和权限表三部分组成，其中，接口描叙了系统模块所调用的系统成员，权限表给出允许用户使用模块中版本的条件。

模块层建立模块的多个版本之间相互依赖关系模型，最常用的是版本树，树中结点表示版本。

版本标志用于区分不同版本，作用是：

- 标识版本；
- 提供查找版本线索，版本标志应隐含版本生成路径。

版本由接口说明、文本和版本记录三部分组成，其中，接口说明给出版本的语义信息，供系统组成平行版本或顺序版本，和进行兼容性检查等用；版本记录记录了版本形成的原因、时间等信息。

用户存取版本时，系统检查和确定所使用的模块，然后按版本模型执行存取动作。

3. 文本层设计

1) 单文件 每个版本以单个文件存放，用于版本文本差别很大情形。

2) delta 模块的多个版本之间具有大量相同文本，如果将每个版本都完整存放，势必浪费大量存贮空间，但按版本形成方式可只存放某完整版本和它版本相对于完整版本的差异delta，由完整版本和delta装配成它版本的完整版本。

版本A早于版本B生成，版本B相对于版本A的delta称为向前delta，反之，版本A相对于版本B的delta称为向后delta。

只要存放所有版本中某个完整版本，用向前delta和向后delta表示其它版本，连接起来组成版本树，就可生成任何版本。假如完整版本取最早版本，其余为向前delta，因此，生成第n个顺序版本利用(n−1)次向前delta，便可得到最新版本。当然这样存放法使回收最新版本所费时间为最长，但一般来说，最新版本使用频率较高，故此存贮方式极不便当；存放最新版本和向后delta时间上为最省，但每条分枝上必须存放完整最新版本，使得存贮开销增大。为此采用混合delta方式，如修改控制系统RCS。

delta由编辑指令组成。delta可交叉存放在一个块里，随着delta数目增多，块也就越来越大，系统搜索必定扫描许多无关的编辑指令，从而影响存取效率。而独立存放delta，版本形成时，系统按delta中指令编辑老版本，这样既节省时间又可方便管理delta。

delta方法适于表示内容相似文本。

3) 布尔表达式　为方便平行版本形式，在程序码中分段插入布尔表达式控制信息，说明该段属哪些平行版本被使用，系统根据布尔表达式顺序取出各代码段形成一个版本，此称为布尔表达式方法。

delta方法与布尔表达式方法亦可结合起来，即，对于顺序版本以delta形式存放；而对于平行版本则施加布尔表达式于一个多版本程序。

4) 索引树　delta方法由两个完整版本计算出delta，版本恢复即为对delta的迭加。

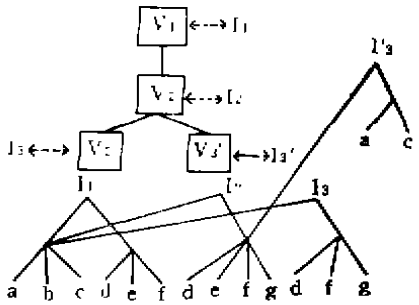

图2　版本模型与索引树

版本由若干数据项即行或块等组成，以数据项索引为结点构成索引树，按中序遍历可得版本全文。索引树构造，新版本中的新数据项构成子索引树，与老索引树中保留的子树合并而得。

版本模型由索引树的根组成，检索按版本模型进行。图2给出版本模型与索引树关系，其中I_1、I_2、I_3、I_3'分别代表版本V_1、V_2、V_3、V_3'的索引树，a、b、c、…、g为数据项，V_1为最初版本，V_2与V_3'平行，V_2由V_1的a、b、c项和d、f、g项组成，V_3'由d、e、f、g、a、c项组成。

利用索引树，增加新版本就是构造索引树，因此版本的合并与添加较为方便，适合于在编辑状态下存取版本；版本亦不需要完整表示，因此可有效用于大型程序版本。

四、结束语

版本管理系统作为一个独立的工具，缺乏对版本自身的语义理解；版本管理的设计主要在文本层，即考虑如何节省存贮空间和提高存取速度。从提供一个集成工具出发，版本管理应与系统的工作有机结合起来。例如，版本管理与操作系统集成使得用户不必区分版本名称和操作系统中的文件路径名，不必使用单独命令对版本进行压缩或恢复，不需文件是否有版本控制，即版本管理对用户透明。

参考文献

[1] Rochkind M. J., The Source Code Control System. IEEE Trans. on Soft. Eng., Vol.1, No.4, Dec., 1975

[2] Tichy, Walter F., Design, Implementation, Evalnation of a Revision Control System. On"6th Inter. Conf. on S. E.", IEEE Sep., 1982

[3] Dewayne E. Perry, Version Control in the Inscape Environment.
@1987 ACM 0270-5257/87/0300/0142