·开发设计·

构件软件配置管理以及其版本控制技术研究

倪晓峰　赵文耘　张　捷
（复旦大学计算机系软件工程实验室，上海 200433）
E-mail：Tonybird333@msn.com

摘　要　该文通过对软件配置管理领域与基于构件的软件开发方法（CBSD）的理论与技术分析，结合 CBSD 的特点、可变粒度构件复用等技术，针对现有软件配置管理工具对 CBSD 支持不足，提出了一个支持 CBSD 的软件配置管理工具 FDSCM，并对其版本控制关键技术进行具体论述，包括构件与构架的版本演化，逆向跟踪和分支合并策略等。

关键词　版本控制　构件　软件配置管理　基于构件的软件开发方法

文章编号 1002-8331-(2005)02-0094-03　文献标识码 A　中图分类号 TP311.52

Component Software Configuration Management & Research on Version Control Technology

Ni Xiaofeng　Zhao Wenyun　Zhang Jie
(Software Engineering Lab, Dept.of Computer, Shanghai 200433)

Abstract: This article describes the theory and technology in the field of software configuration management and component based on software development (CBSD). Aimed at the lack of supporting CBSD with today's SCM tools, a new SCM tool FDSCM which fully supports CBSD is presented, we also discuss the version control technology including the version evolution of components and component structures, converse track, branch and merge strategy.

Keywords: version control, component, Software Configuration Management (SCM), CBSD

1　引言

经过多年软件开发方法及技术研究，软件开发的质量及效率有了较大的提升，但是依然存在两个问题，一是随着软件规模的增长，开发人员的流动，软件的维护将变得越来越困难。二是如何进行大规模软件生产。配置管理的出现很好地改善了软件维护问题，基于构件的软件开发方法（CBSD）也被提出用来解决软件复用问题，从而能够进行大规模的软件生产。因此，CBSD 与软件配置管理（SCM）两者的结合将变得尤为重要。

1.1　定义

（1）软件配置管理：简称 SCM，是软件过程的关键要素，是控制软件系统演化的规程。它包括标识给定时间点上软件的配置，系统的控制配置更改，维护配置在整个生命周期中完整性和可跟踪性[1]。它有下面一些概念

（2）配置项/配置：是处于配置管理控制之下的对象[5]，配置项是配置管理中最基本的单位，配置是配置项的选择性集合，它与配置项的关系也就是简单的包含关系。

（3）基线：是对象成为新配置项的时间点[5]，通过基线的定义可以同步并发和发布产品，有时也把基线称为里程碑。

（4）版本控制：是 SCM 的核心部分，是对开发过程中配置的版本进行记录和保存，通过合适的规则来创建新版本，删除旧版本，合并已有版本等，提供一种策略来选择版本创建基线[6]。其中牵涉到以下一些概念：

（1）版本、版本项：版本代表了一个演化对象的状态，版本项是版本所代表的实体。

（2）增量：多个版本会共享版本项的某些部分，对于版本的保存只需要存储不同版本间的差异，而这种差异就是增量。

（3）版本规则：创建，删除，合并，选择版本的策略。

（4）粒度：版本项的范围，是文件还是目录，或是两者都有。

（5）工作区和事务处理：工作区是进行并行开发的开发者的工作场所或存储空间，它被广泛应用于检入/检出模式。事务是一段时间内的版本变化，事务的特点在于整体性，也就是版本的变化要么全变要么都不变。

（6）基于构件的软件开发方法：简称 CBSD，是一种新兴的软件开发方法，是根据软件复用研究结合软件构件技术发展而产生的[2]。CBSD 认为可以通过选择合适的商业构件（COTS），并把其组装成为一个定义明确的软件构架来生成软件系统[3]。

（7）构件：构件是一个独立发布的功能部分，可以通过它的接口访问它的服务[4]，在 CBSD 中构件就可以分为两类，原子构件（primitive component）和复合构件（composite component）[2]。

①原子构件：是 CBSD 中最小的逻辑单位，通常由一组文

基金项目：国家 863 高技术研究发展计划项目：上海构件库及其应用（编号：2002AA114010）；上海市科技攻关计划项目：可变粒度的软件配置管理工具研究（编号：035115026）

作者简介：倪晓峰（1980-），男，硕士研究生。研究方向：软件工程。赵文耘（1964-），男，教授，博导，研究方向：软件工程。张捷（1980-），男，硕士研究生，研究方向：软件工程。

件组成[2],它们之间没有关系,只是简单的叠加构成一个集合,对外提供服务的同时也可以请求其它构件的服务。

②复合构件:是由原子构件或复合构件组成的构件[2],组成它的构件之间有较强的联系,在不同的系统中,复合构件可以有也可以没有实体。

③构架:是构件的组合方式的描述[4],一般来说构架就是一个没有实体的复合构件。

1.2 CBSD 对配置管理的新要求

CBSD 对配置管理提出了 4 个方面的要求,包括:配置识别,版本控制,变更控制和协同开发[2]。

(1)配置识别:配置识别主要解决那些可以成为配置管理中的管理对象(配置项)和如何来选择配置项组成配置的问题。CBSD 中主要接触到的是构件和构架,这时候的配置项就应该是原子构件,复合构件,和构架。

(2)版本控制:版本控制可以说是配置管理中核心的部分,CBSD 要求对不同的配置项有不同的版本控制策略,对于原子构件来说,物理上它是由一组文件构成的,它的版本演化不能完全根据其组成的文件,要有选择性。复合构件的版本变化应包括两方面,一是构架本身的变化(结构的改变和原子构件的替换),二是组成成分(原子构件)的版本演化也要影响它的版本。

(3)变更控制:变更控制主要是跟踪整个软件开发过程中的变化,记录相关信息,日志等。构件的有结构性对变更控制提出了新的要求,即一致性,完整性检查。

(4)协同开发:在 CBSD 中,开发人员被分为两类,构件生产者和构件消费者,配置管理需要协调好构件生产者与构件生产者之间,构件消费者与构件消费者之间,尤其是构件生产者与构件消费者之间的协同开发。

1.3 SCM 的现状与其在版本管理技术上的不足

至今已经有很多厂商和实验室推出了自己的 SCM 产品或原型,如 VSS,ClearCase,JBCM 等,但是这些工具或对 CBSD 没有支持,或者支持不够,都没有满足 CBSD 对配置管理工具的新要求。

VSS,TeamSource 和 CVS,这些都是较简单的低粒度的文件版本管理工具,对 CBSD 无相应支持,它的版本创建只能建立在最新版本之上。

北大青鸟的 JBCM 将构件引入到配置管理系统中。JBCM 有效提升了版本管理粒度。但是它的构件是没有结构的,即没有构架的演化,复合构件的版本演化会引起版本膨胀,对于 CBSD 没提供有力支持。

ClearCase 和 Rational 的其他一系列工具紧密集成,提出了两级工作区的概念模式,可以有效地控制非构件版本演化中的膨胀问题。但 ClearCase 并没有对 CBSD 提供支持。高度集成性、复杂性使得 ClearCase 在企业中的实际应用非常困难。

针对以上的情况,从企业迫切需求和新技术发展要求出发,在充分吸收现有研究成果的基础上,着手研究并设计了一个完全支持 CBSD 的 SCM 工具 FDSCM。

2 支持 CBSD 的 SCM 工具–FDSCM 的总体结构

FDSCM 是一个完全支持 CBSD,满足 CBSD 对配置管理提出 4 点要求的 SCM 工具,在 FDSCM 中管理的对象是构件,每一个构件都是有结构的即构架,构件与构架的版本是独立演化的。FDSCM 不仅仅可以被应用于 CBSD,对于当前的其他开发方法如 OOP 也提供同样的支持,它使用一种逆向分析的手段可以通过对遗留系统分析来获取各种粒度的构件进行管理。FDSCM 有以下特点:

(1)构件与构架的版本独立演化并支持从任意版本开始的版本演化。

(2)智能化的双向的版本跟踪和逆向分析。

(3)体现应用族、构件系统和应用系统的开发方法。

(4)提供与基于构件开发相适应的过程管理。

(5)同时支持构件开发和构件使用。

2.1 FDSCM 的总体结构图(图 1)

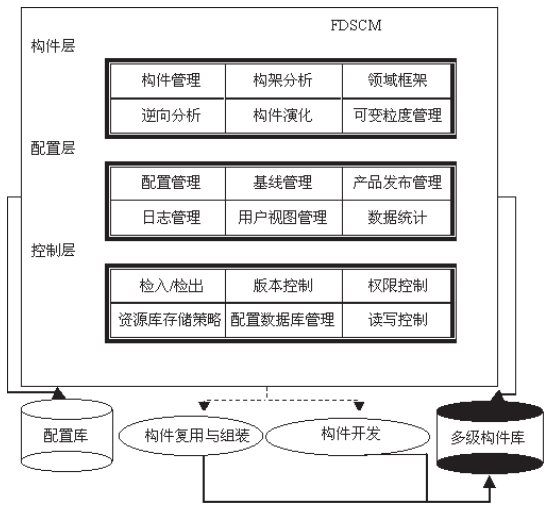

图 1 FDSCM 的总体结构图

构件开发和构件复用与组装,使用的是多级构件库。FDSCM 对这两类活动都支持,与之关联的配置库和多级构件库。FDSCM 分为三层:控制层主要关注的是数据库存储,资源库存储,检入/检出策略和版本控制等底层核心技术。配置层关注的是配置管理,基线管理等,还包括构件形成配置的策略。构件层包括系统构件管理,构架分析,逆向分析等与构件相关的服务。每一层都请求下一层服务,同时也为上一层提供服务。

3 版本控制关键技术研究

版本控制是软件配置管理中的关键部分,现在的配置管理工具中存在很多的版本控制方法,这些方法的区别主要在于,版本控制的对象(文件,目录),版本的组织方式(版本树或者多维的版本空间),版本粒度,版本变化的侧重点(基于状态的还是基于变化的)和版本选择策略等等。在 FDSCM 中构件是有结构的,构件与描述构件的构架在版本演化过程中是独立的。下面的论述给出一个适应 CBSD 的版本控制技术以及设计思想。

3.1 版本控制技术的结构图和对 CBSD 支持

版本控制主要有以下几方面的任务:版本信息的存储,版本的创建,删除,合并,版本的选取,分支合并管理,版本表式,版本审计,工作区管理和事务管理。FDSCM 的整体版本控制结构图图 2 所示。

版本控制是由三个部分组成:版本控制驱动是核心,通过它来完成包括版本创建、删除、合并,版本选取,分支合并,版本审计等。事务管理主要负责对长时间的修改所引起的版本变化进行管理。工作区管理主要是为了支持协同开发。

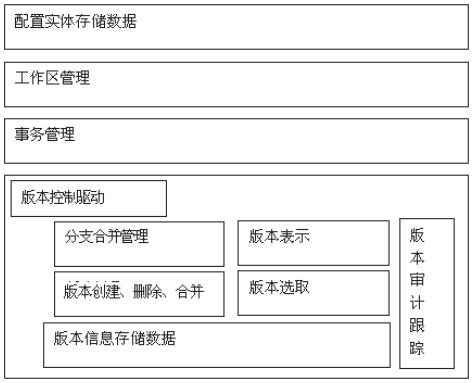

图 2　FDSCM 的整体版本控制结构图

众所周知，在配置管理工具中，版本控制就是对配置的版本演化进行管理，配置的基本单位是配置项，所以对配置的版本管理归根结底是对配置项的版本管理。在版本管理中，把配置项分为三类：原子构件配置项 CP、构架配置项 CF 和复合构件配置项 CC。定义配置项 CI=CP|CF|CC，配置 C=(CI)⁺。

3.2　版本控制驱动

版本控制驱动是版本控制的核心，它负责版本信息存储，版本的维护，分支管理，和版本审计等。

3.2.1　版本信息存储

版本是版本项的状态，版本信息应该与版本项分开存储，事实上笔者也做到这一点，把版本信息存放在配置库中，而版本表示的实体存放于项目构件库中。配置项的存储有三种方式：直接存储，增量存储，倒增量存储。

对于 CBSD 而言，原子构件在物理上是一组文件的集合，如果因为某一个文件的版本演化引起的原子构件版本演化，采用直接存储难免太浪费空间，所以使用文件粒度倒增量存储模式。这种模式的最小存储单位是文件，也就是说对于文件采用的直接存储，而构件和构架就是增量存储。例如，构件 B 由 10 个文件 $f1,f2,\cdots,f10$ 构成，B 的初始版本为 1，一次修改后 f1 变化了，而其他文件都没有变化，那么 B 的 2 版本就是 $\Delta f1,f2,\cdots,f10$，其中 $\Delta f1$ 是 $f1$ 修改的完整实体。对于构架配置项的版本演化来说，因为构架是描述构件的组成结构，增量对它来说就是直接存储。对于复合构件配置项，构件的版本变化影响它所组成的复合构件的版本，这里仅需要保存变化构件的版本而其他没有变化的构件都不变。

3.2.2　版本号的演化算法

在 FDSCM 中构件是有结构的，构件的结构是通过构架来描述的。构架有自己的版本演化过程，与组成构件的实体版本演化完全分离。构架与实体演化的独立性可以很好地解决版本膨胀的问题。

配置版本号(V)由三部分组成：父版本号(Vf)，分支号(Vb)，分支版本号(Vv)，V=Vf+'-'+Vb+'.'+Vv。如图 3。

FDSCM 是支持从任意版本开始演化的，版本号的生成比较复杂，它的生成算法如下：

(1)版本检出时记录检出版本号 Vout，返回记录号 r。

(2)版本检入时根据记录号 r，查询检出版本号 Vout。

(3)根据检出版本号 Vout，得到此版本在版本树上的节点 Tv。

(4)根据 Tv，得到 Tv 的儿子节点集 Tvc[]

(5)如果 Tvc[]为空，则版本为当前分支上最新，新版本号 Vnew 等于 Vout 最后一位数字加 1。

(6)否则 取 Tvc[]中的最大分支号 Fmax，新分支号 Fnew=Fmax+1，新版本号 Vnew=Vout+'-'+Fnew+'.'+1。

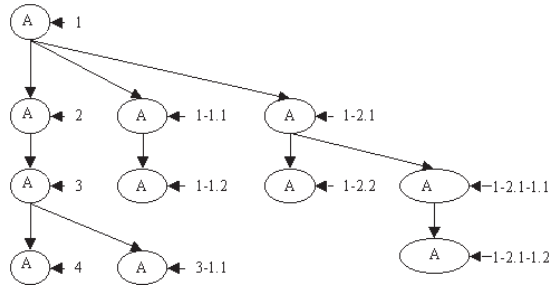

图 3

3.2.3　版本创建、删除、合并策略

版本的创建在 3 个时间点发生：

(1)在创建新配置项的时候，版本就为最初版本 1。

(2)在配置项被检入时发生，版本可能朝纵向或横向演化。演化规则是：

①如果配置项是被互斥写检出的，那么版本朝纵向演化。

②如果配置项是被共享写检出的，那么版本朝横向演化。

③如果配置项是被只读检出的，那么版本不演化。

(3)在分支的时候，配置项的分支是版本朝横向演化。

版本控制方法中有三类配置项(对于不同的配置项，版本创建是不同的)：

(1)原子构件配置项：因为原子构件的版本变化，需要触发构架的版本变化，原子构件配置项的新版本，要引发包含此构件的所有构架的新版本创建。但是这样的方法会引起构架版本的迅速膨胀，所以对此进行的改进，类似构件与组成构件的文件版本变化关系，在系统中构架的版本与构件的版本单独演化，当构架的版本除了结构变化时需要创建新版本以外，另外的版本变化是通过对其中构件版本的选择产生。如构架 F 版本 1 包含构件 C1 1,C2 1,C3 1，当 C1 从 1 演变为 2 时 F 版本不变，经过多次演变后变为 C1 3-1.1,C2 4,C3 1-1.1。这时如果构架的版本需要发生变化时，它可以选择 C1,C2,C3 的版本。可能 F 2 版本包含了 C1 3,C2 1,C3 1,也就是说构件版本的变化只是通知其被包含构架，构架可以选择版本变化或不变，这样可以有效避免构架版本膨胀的问题。还有一方面就是原子构件与原子构件之间的调用关系，比如构件 A 使用了构件 B 那么当 B 发生版本变化时，A 也要发生版本变化。关于这一点笔者还在研究中。

(2)复合构件配置项：对于复合构件的检出，主要是为了修改构架的结构或者构架的组成成分，一般不涉及到构件的修改，所以无论复合构件配置项是如何检出的，其包含的构件都是只读检出，所以复合构件的新版本不影响构件。

(3)构架配置项：构架配置项的版本是与构件的版本分开演化的，构架配置项的版本修改只对自己有影响。

版本的删除与合并，版本是不允许手工删除的，版本的删除只发生在版本合并时，版本的合并也有两种：纵向和横向。纵向的合并发生在工作区中，横向的合并，发生在分支合并中。对于不同的配置项，也有不同的合并策略：

(下转 145 页)

硬件模块，如处理器与智能传感器，使得系统功能实现直接以硬件方式实现，进一步完善网络控制系统的 Matlab 环境功能。（收稿日期：2004 年 6 月）

参考文献

1.Michael S Branicky，Stephen M Phillips，Wei Zhang.Scheduling and Feedback Co-Design for Networked Control Systems[C].In：IEEE Conf on Decision and Control，Las Vegas，2002：10~13

2.Anton Cervin.Integrated Control and Real-Time Scheduling[D].PhD thesis.ISRN LUTFD2/TFRT--1065--SE，2003-04

3.Storch M F，J W-S Liu.DRTSS：A simulation framework for complex realtime systems[C].In Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium，1996

4.Dan Henriksson，Anton Cervin.TRUETIME 1.1-Reference Manual[R].Department of Automatic Control Lund Institute of Technology，2003-03

5.M Marcos，J Portillo，J M Bass.Matlab-Based Real-Time Framework For Distributed Control Systems[C].In：Proceedings volume from the IFAC Workshop AARTC'2000，Palma de Mallorca(Spain)，2000-05

6.Jad El-khoury，Martin Törngren.Towards a Toolset for Architectural Design of Distributed Real-Time Control Systems[C].In：22nd IEEE Real-Time Systems Symposium(RTSS'01)London，England，2001-12

7.王智，王天然，Ye-qigong SONG 等.工业实时通讯网络(现场总线)的基础理论研究与现状(上)[J].信息与控制，2002：4

8.Palopoli L，L Abeni，G Buttazzo.Realtime control system analysis：An integrated approach[C].In：Proceedings of the 21st IEEE Real-Time Systems Symposium

（上接 96 页）

（1）原子构件配置项：纵向就以最新版本作为新的版本，其他版本删除，包含此构件旧版本的构架版本删除。横向通过人工选择等到新版本，合并的版本都删除，包含此构件的旧版本的构件版本删除，产生新构件版本。

（2）复合构件配置项：纵向就以最新版本作为新的版本，其他版本删除，横向通过人工选择等到新版本，合并的版本都删除。

（3）构架配置项：构架配置项的版本修改只对自己有影响。

3.2.4 分支合并管理

分支合并管理主要是用来支持协同开发的，在 CBSD 中，由于开发的过程比较简单而且标准性较强，所以采用的整体分支模型是基于任务的分支模型[7]，这个模型是基于目的的分支模型的一种演化，开发总是基于某种目的的，开发人员总有某个任务才进行开发的，为每个任务而去对整个项目或配置进行分支的方法就是基于任务的分支模型。

在基于构件的开发中，协同开发不仅仅是构件使用者之间、构件开发者之间的协同开发，还有构件生产者和构件消费者之间的协同开发。因为 FDSCM 对构件的开发和构件复用两者都提供支持，企业级的构件开发是独立于任何项目的，通过基于任务的分支模型，对一个构件可以创建两种分支，一是生产分支，二是消费分支。生产分支之间，使用分支之间的合并与其他的分支模型一样，对生产分支与消费分支的合并就有新的策略，因为开发构件的实体是存储在企业级构件库中，而使用构件的实体是保存在项目构件库中，两者的合并不仅仅是版本的合并，还需要进行构件实体的更新。

3.2.5 版本审计、跟踪

版本的审计、跟踪主要是为了配合变更管理，用户可以通过版本跟踪得到版本的修改信息，包括版本增加，删除和合并。版本的审计是版本的历史记录，尤其是对构架的版本记录，可以方便地对其构件的版本进行跟踪查询。

在 FDSCM 中对构件的版本变化采用双向的跟踪策略：正向主动：当结构发生变化时，无论是结构变化，还是组成部分发生替换，都会引起相应部分的版本变化。逆向被动：当构件、构架升级时（这里主要是构件生产者引起的变化），正是因为 FDSCM 同时支持构件开发和构件复用的，所以你们可以分析所有使用了此类构件或构架的应用，通知其组成的构件发生版本变化。

3.3 事务管理和工作区管理

事务主要是支持需要保持一致性的连贯修改，和其版本演化。这里主要通过基于活动的分支模型和两级工作区管理来完成对事务的支持，把属于事务视为一个活动集，对事务的处理就是对活动集的处理。

工作区管理采用的是两级工作区管理模式。例如：构件开发者检出构件 C 版本 2 进行开发，在检入之前 C 所有的版本变化都在工作区内完成，当 C 检入时，构件开发者通过版本合并来得到最新的版本检入。这样在工作区中存在的版本缓冲对于构件使用者是透明的，而开发过程中又是存在的，能很好地利用配置管理工具进行管理。

4 结语

FDSCM 是一个完全支持 CBSD 的 SCM 工具，它的版本控制技术通过构件与构架的独立演化解决了版本膨胀问题，通过双向跟踪支持构件的变更管理，通过基于活动的事务管理的多级工作区管理来达到协同开发。FDSCM 为企业使用 CBSD 提供了有力支持，现在笔者已经完成了原型设计，将在复旦天翼计算机有限公司范围内使用并逐渐完善。以后的工作将会集中在 FDSCM 的性能改进，遗留系统的构件提取以及企业分布式应用上。（收稿日期：2004 年 7 月）

参考文献

1.IEEE Std 610.12-1990.IEEE Standard Glossary of Software Engineering Terminology(ANSI)[S]

2.梅宏，张路，杨芙清.基于构件的软件配置管理模型和支持系统[J].J Comput Sci & Technol，2002：17(4)

3.Xia Cai，Michael R Lyu，Kam-Fai Wong et al.Component-based software engineering technologies development frameworks and quality assurance schemes，IEEE 2000：1530~1362

4.Alan W Brown 著.赵文耘，张志等译.大规模基于构件的软件开发[M].北京：机械工业出版社，ISBN 7-111-11918-5，2003-07

5.James E Tomayko.Software Configuration Management，SEI Curriculum Module.SEI-CM-4-1.4 Software Engineering Institute CMU，1990

6.Bernhard Westfechtel，Bjùrn P Munch，Reidar Conradi.A layered architecture for uniform version management[J].IEEE TRANSACTIONS ON SOFTWARE ENGINEERING，2001：27(12)

7.Chuck Walrad Darrel Strom.The importance of branching models in SCM.0018-9102，IEEE，2002