第41卷
2005年10月

南京大学学报(自然科学)
JOURNAL OF NANJING UNIVERSITY
(NATURAL SCIENCES)

Vol.41
Oct., 2005

一个支持构件演化和变更管理的配置管理工具*

彭 鑫,赵文耘,吴毅坚
(复旦大学计算机科学与工程系软件工程实验室,上海, 200433)

摘 要: 配置管理可以为软件产品的有序演化提供保障,而基于构件的软件开发(CBSD)对配置管理提出了新的要求.介绍了一个新的配置管理工具FDSCM,并对该工具在版本管理、变更管理和构件演化支持三个方面对CBSD的配置管理支持进行了探讨,重点论述了变更管理与版本管理的集成、构件的版本管理以及配置管理对构件开发视图和复用视图的支持等关键技术.

关键词: 配置管理,版本管理,变更管理,构 件,构件库,CASE工具
中图分类号: TP 311

A SCM Tool Supporting Component Evolution and Change Management

Peng Xin, Zhao Wen-Yun, Wu Yi-Jian
(Department of Computer Science and Engineering, Fudan University, Shanghai, 200433, China)

Abstract: Configuration Management (CM) ensures the orderly evolution of software artifacts. Component-Based Software Development (CBSD) presents new requirements for CM. This paper discusses CM for CBSD from three aspects of version management, change management and component evolution supports. The focuses of this paper are integration of change management and version management, version management of components, supports of CM for the development view and reuse view of components.

Key words: configuration management, version management, change management, component, component library, CASE tools

软件配置管理在软件开发的诸多支持活动中处于核心地位,主要关注于记录并控制软件开发过程所产生的全部资源的演化[1].软件配置管理包括一整套严格的规范和技术,用于在开发过程之中以及开发过程之后发起、评估和控制软件产品变化,强调配置控制的重要性[2].随着软件系统规模和复杂度的不断增大,配置管理工具的重要性也得到了越来越广泛的认同.目前已经有许多成熟的配置管理工具,其中比较流行的有Rational的ClearCase[3,4]、微软的VSS、开源的CVS等.国内也有一些软件厂商或研究机构推出了自己的配置管理工具,例如JBCM[5]和Firefly[6]等.这其中ClearCase的市场占有率最高,功能也十分强大,但其易用性也较差.

现有的配置管理工具在实际应用中仍然存

* 基金项目:国家863计划(2004AA112070,2004AA113030),国家自然科学基金(60473061),上海市科委科技攻关项目(035115026)

在一些困难.一方面,变更管理[7]需要与配置管理结合起来提供更加系统的演化控制,只有这样变更管理的功能才是系统和完整的.另一方面,基于构件的软件开发正在成为新的软件开发范例[5].这种全新的开发方式对配置管理也提出了新的要求,其中最主要的是对构件演化的支持.为此,我们在国家自然科学基金和上海市科委项目资助下开展了配置管理工具方面的研究,并开发出了一个新的配置管理工具FDSCM.

与其它配置管理工具相比,FDSCM主要有以下3个特点:

(1) 两级版本管理:本地级和服务器两级版本管理,为开发者提供两个层次的开发视图.

(2) 与变更管理系统紧密结合:变更请求处理流程中每一个步骤都可以引发一次开发活动,从而使变更请求与活动中的版本演化关联起来.

(3) 支持构件独立演化:构件成为一类独立于特定项目的版本实体,在构件系统中进行演化,构件的开发者和使用者分别从不同的视图对构件进行维护或复用.

主要从变更管理与版本管理系统的集成以及对构件化开发的支持等方面展开研究.本文首先介绍FDSCM的版本管理策略,然后介绍了变更管理系统以及与版本管理的结合.构件演化支持是研究重点.

1　版本管理

版本管理是配置管理工具的基本功能,不同的配置管理工具其实现策略也不同:CVS主要支持基于目录和文件的检出/检入;ClearCase[3,4]以版本对象库(VOB)、视图(View)和工作空间(Work Area)这三个层次进行版本控制;JBCM[5]将构件(原子构件/复合构件)也作为一类版本实体,提高了版本管理的粒度.

FDSCM版本管理的主要特点是本地级和服务器两级版本管理.JBCM以子项目作为项目结构单元.FDSCM与此类似,使用层次构造的子系统组织所有的配置项.子系统是开发任务和权限分配的单位,项目是最高一级的子系统,配置项是基本的版本实体.原子配置项由文件和目录组成,而复合配置项还可以由其它配置项组合而成.构件也是一类版本实体,构件的版本管理将在第3部分中详细介绍.

1.1　两级版本管理　开发者通过团队协作参加项目开发时具有多种视图.在项目级上,每个开发者按照角色和所承担的任务提交符合一定要求的产品.在本地级,开发者为了完成开发任务可能需要反复修改某些文件或者创建一些辅助的文件,这些文件的版本对于本次本地开发活动来说是有意义的,但对于项目级开发有意义的只是其中一部分,这部分版本才需要提交到服务器端.因此,开发者在本地具有个人开发视图.

许多配置管理工具都支持本地工作区,但都不提供本地版本管理支持.Firefly[6]的本地工作区支持脱机操作,但只能记录脱机历史以在下次联机时提交,并不能提供本地版本管理支持.ClearCase通过工作空间(Work Area)为用户提供私有空间[3],工作空间通过开发视图与版本对象库(VOB)交互.

FDSCM的版本管理框架如图1所示,包含两级版本管理:本地级和服务器级.本地版本管理在个人工作区上开展,通过本地配置信息库和资源库进行版本管理,对于其他开发者是不可见的.活动是本地版本演化的载体,任何本地版本活动都必须在活动下进行.活动属于当前用户参与开发的一个项目.活动中可以新建配置项,也可以从服务器上下载该项目的配置项.本地检出/检入发生在活动和本地配置库之间,配置项的本地演化体现为本地版本的增长.当本次开发活动结束时,活动中某些配置项版本被提交到服务器完成服务器端的版本演化.

本地开发具有较低的粒度,因此本地版本管理的单位是配置项和文件.本地配置项所包含的所有文件都具有本地版本,可以以文件为单位进行检出以及多个版本的文件比较等.而服务器端版本管理的粒度较高,基本单位是构

件和配置项.这样的两级版本管理既满足了不同层次的开发需要,又可以大大减少服务器端的版本冗余.

1.2 集成基线 开发者在完成个人开发任务时,除了需要创建或修改的配置项,往往还需要其它资源辅助开发,例如用来构造集成开发环境.FDSCM提供集成基线来方便这类资源的下载,每个子系统都可以创建集成基线.基线是相关元素版本的一个具有里程碑意义的组合[4].在FDSCM中,集成基线是一系列相关配置项特定版本的组合(如图1).集成基线的下载是只读的,其中所包含的配置项不允许修改后再提交到服务器.集成基线为小组中所有的用户提供一个共同的阶段性的开发起点,其中可以包含完成系统编译和运行所需要的代码文件、相关的文档等.当小组结束一个阶段的开发后,子系统配置管理者可以创建新的集成基线,提升小组的整体开发进度.

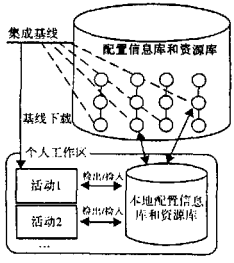

图1 FDSCM版本管理框架

2 变更管理

软件变更管理必须满足变化控制和变化追踪这两方面的需求[7].一个变更从提出到处理完毕包含了一系列的评估、批准、任务分配、实施、审核确认等步骤.从配置管理的角度看,变更生命周期主要包括审核决策和实施两部分.审核决策在开发小组外部进行,主要体现为一系列文档的处理过程.而实施部分则体现为一系列开发活动和版本演化.

2.1 基于Web的变更生命周期管理 在FDSCM中,变更管理流程主要体现为变更生命周期模型上一系列文档的处理过程(图2).FDSCM支持用户自定义变更生命周期模型,其中的每一个步骤由特定的角色负责.用户按照自己所承担的角色负责一个或几个步骤上的变更请求处理:接收变更请求以及相关文档,根据具体情况决定下一步的处理方向并按照所选择的处理方式填写相应的文档.例如在图2中的变更处理模型上,负责"审核分配"的用户将决定是否决该变更请求(直接转向"结束变更")还是分配给相应的开发者进行实施(转向"实施变更").在这两种情况下该用户需要分别按照预定义的文档格式在Web页面上填写变更文档,例如这两种文档可以分别被命名为"变更请求否决说明"和"变更实施说明".通过这种方式,变更请求在整个生命周期中的处理以及相应的文档可以处于系统的控制和记录之下.目前我们的变更管理系统还支持变更处理情况的查询、统计,变更请求的整个生命周期中的处理情况以及每一个环节的文档可以一起倒出,为用户提供完整的变更处理历史记录.

2.2 变更管理与版本管理 通过这样的处理流程,每一个变更从提出到结束的处理信息、文档都可以得到控制和追踪.但用户往往还特别关心另一个问题:变更处理过程中产生了哪些新版本,工作量如何,是否能够建立版本与变更请求之间的追踪关系.由此可见,变更处理必须与相应的版本演化联系起来.Firefly[6]采用面向任务(即变更)的管理模式,任务可以关联到一个变更集,这个变更集可以一起提交.论文[7]提出基于过程管理和配置管理进行变更管理.在FDSCM中,变更请求通过活动与版本管理相联系.如图2,变更请求(多个变更请求可能合并后处理)进入实施阶段后分配给特定的开发者并开放相关权限.该活动将成为开发者的一个本地活动.经过一系列的本地版本演化后,开发者从本地版本中挑选一些作为开发结果一起提交,经审核后结束本次活动.这样,活动中所发生的版本演化就与变更请求关联起

来,建立起变更与版本之间的追踪关系.通过自动的结构分析和文件比较,FDSCM 还能计算相应的工作量(例如时间、增加或修改的代码行数等),为变更影响分析提供了支持.

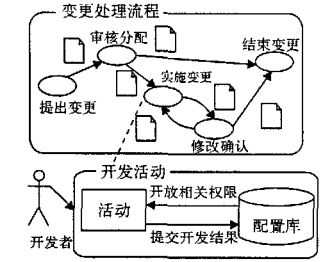

图 2　变更管理与版本管理

2.3　变更的分支与合并处理　变更请求的处理过程并不总是线性的,例如同时提出的几个变更可能是重复或相关的,可以合并处理,或者某个变更请求需要分解为多个部分分别处理.FDSCM 支持变更请求的合并和分解.分解可以发生在任何非起点或终点的步骤上,用户通过分解操作为变更请求创建新的处理分支.这些分支的后续处理相互独立,但它们此前的处理流程共同继承自原来的变更.合并操作与此相反,由用户将多个变更请求合并处理,此前的多个变更的相关信息都能在此后的变更处理中获得.

3　构件演化支持

许多软件企业已经逐渐在向基于构件的软件开发(CBSD)方式转变.在 CBSD 中,构件组装是主要的系统构造方式[5].构件的开发、维护和复用涉及到一系列配置管理活动.在 CBSD 中,构件库[8]是企业的核心资产以及构件开发和复用的核心.如何在构件库固有的构件分类、组织、存储和检索[8]等功能的基础上增加配置管理支持是一个关键问题.论文[9]提出了一个构件库的配置管理模型来支持构件的版本和配置管理.这个模型与特定的构件库相关,且关注于黑盒构件.论文[10]在构件描述语言(CDL)上扩充系统演化信息,使构件组装工具与配置管理系统结合起来.配置管理系统和构件库具有不同的关注点.在 FDSCM 中,我们通过建立配置管理系统中的构件与构件库中构件之间的映射关系来区分构件的开发视图和复用视图(图 3),从而为构件演化提供支持.

3.1　构件版本管理　一般认为构件构件是指应用系统中可以明确辨识的构成[11].从更加严格的意义上说,构件还应该完成一定的功能、具有良好的封装.我们从配置管理的角度出发,认为独立演化也是构件的一个很重要的特性.一个构件的可复用性来源对一系列问题的抽象和共性的提取,使构件可以在多个项目中复用.因此构件如果随着特定项目发生演化,那么很容易因特化而失去原有的一部分可复用性,而且版本数量也会急剧增长.

FDSCM 中的构件是一类独立的版本实体,分属不同的构件系统,例如按功能划分的 UI 构件系统、报表构件系统等.如图 3,构件系统可以视为永不结束的"项目",构件在构件系统中演化,由相应的构件开发者进行构件的开发和维护工作.普通的项目用户可以检索并下载构件到项目中成为一个配置项,但没有修改的权限,也不需要了解构件的实现细节.如果对构件有变更请求可以向构件开发者提出,由他们完成构件的维护工作.通过这样的方式,构件开发者可以通过对项目开发者要求的总结和提炼获取构件的共性要求,从而保持构件的独立演化性.

3.2　构件的开发视图和复用视图　基于构件的软件开发必须区分构件的两个视图:开发视图和复用视图[9].构件的开发视图多数情况下还是使用传统的开发方式,例如原子构件通常实现为一组文件.构件的复用视图才是基于构件的开发,例如使用构件组装目标系统.

构件的复用者关心的是构件所体现出来的功能、性能以及复用条件等,并不关心构件开发和维护的细节.构件库面向构件复用者,提供构件的分类、组织和检索等功能.因此构件库属于

构件复用视图.另一方面,构件的开发者负责构件的开发和维护,包括创建新构件、修改 bug、增加功能、满足构件变更要求等.因此构件的开发视图需要维护构件的各个版本、演化历史、与需求设计文档的追踪关系等,与一般的版本管理类似.

如图 3,项目的开发者使用构件库的构件检索接口进行构件检索并将构件下载到项目中.而构件开发者在构件系统上进行开发活动.构件的开发与项目开发类似,也是通过本地工作区上的活动向服务器提交开发结果,不同的是这里将所在的构件系统作为"项目".构件系统将构件作为版本对象管理,维护构件的所有历史版本以及各种版本关系.通过建立构件系统与构件库之间的映射关系来实现构件的开发和复用视图.图 3 中,构件 A、B、C 分别对应构件系统中的构件版本.其中,B 和 C 对应于构件系统中同一个构件的两个版本.因为这两个版本具有不同的可复用性,因此在构件库中映射为两个构件.作为复用者的项目开发者只关心这两个构件所体现出来的不同的可复用性,而不用关心它们的开发历史.对于构件的开发者来说,这种版本信息却是至关重要的.例如如果构件 B 中发现了 bug,那么构件开发者可以知道构件 C 可能也会受到影响.从某种程度上说,构件库对应于项目开发中的产品发布,因为构件开发目的就是提供可复用构件.从图 3 中可以看到,构件系统中的构件通过发布过程向构件库提交.根据构件的不同类别,发布的方法和内容也不一样.例如源代码构件发布的是源代码,而二进制构件发布的是 Build 后得到的二进制构件.

构件库中除了存储构件实体外,还需要提供构件的描述信息,包括接口描述、构件间关系等.构件库中的构件关系主要是不同构件间的依赖、引用、特化等关系等[12].这些描述信息有许多都可以从作为开发视图的构件系统中获取.因此,构件系统向构件库发布的除了构件实体外还包括相关的描述信息.总之,构件系统为构件库提供构件演化支持,通过构件映射关系将开发视图(构件系统)中的构件信息经过过滤和重组转换为复用视图(构件库)中复用者所关心的各种信息.

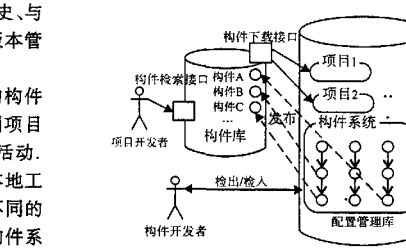

图 3 构件的开发视图和复用视图

3.3 构件的结构和实体 在 CBSD 中,存在两种构件:原子构件和复合构件[5].原子构件实现为一组文件,而复合构件由一组相关的其它构件组合而成.对于严格意义上通过组装获取的复合构件,参与复合的子构件是按照特定的结构(构架)组合在一起的.如图 4,复合构件 CC 的 1.0 版在构架 Arch 的 1.0 版上组装了 A、B、C、D 四个子构件的某个版本.由图中可看出,复合构件的版本取决于两个方面:构架版本以及构架上组装的各子构件版本.CC 的 1.0 版和 1.1 版的差别在所组装的 A 和 D 的版本不同,而 1.2 版则使用了构架 Arch 的 1.1 版.

在 FDSCM 中,构架也是一类版本实体(如图 4 中的构架 Arch).因为构架并不从属于特定的构件或系统,也是一类单独演化、可独立复用的软件资产.构架的版本实体可以用构架描述语言(ADL)来表示.复合构件配置的一致性可以通过构架和构件描述进行,例如配置中所选择的子构件版本是否满足接口要求等.通过

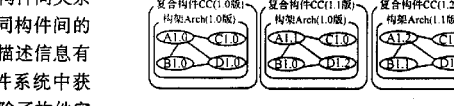

图 4 复合构件、原子构件与构架

这种配置管理方式,构架的开发和维护就可以独立进行,从而使更好地适应 CBSD 的要求.

4 总　结

本文主要针对基于构件的软件开发方式下配置管理的特点,研究了版本管理对变更管理以及构件演化的支持.在此基础上本文结合配置管理工具 FDSCM,从版本管理、变更管理和构件演化支持 3 个方面详细介绍了所涉及的各种问题.其主要创新点在于:提出并实现了两级版本管理;基于本地版本管理中的活动将变更管理与版本管理结合起来;在构件版本管理方面提出结构与实体单独演化,使构架成为一类可独立演化和复用的版本实体;在构件演化支持方面提出并实现了构件库与构件系统之间的映射结构,使得构件的开发视图与复用视图相分离.

目前 FDSCM 的开发已经完成,并在三家软件开发企业中得到应用.我们下一步将继续在构件变更影响分析以及构件开发、构件复用的过程支持等方面进行研究和开发.

References

[1] Lindsay P, MacDonald A, Staples M, *et al*. A Frameworks for Subsystem-based Configuration Management. Software Engineering Conference. Proceedings, Australian,2001,275～284.

[2] James E T. Software Configuration Management. Technical Report. SEI Curriculum Module SEI-CM-4-1.4, 1990.

[3] Shang W R. Software Configuration Management and ClearCase. Modern Computer, 2003, 173 (10): 14～16.(尚巍然. 软件配置管理与 ClearCase. 现代计算机, 2003, 173 (10): 14～16).

[4] Jim. Rational ClearCase LT Users Guide. http://www. 8848software. com. (Jim. Rational ClearCase LT 使用指南. http://www.8848software.com).

[5] Mei H, Zhng L, Yang F Q. A Component-Based Software Configuration Management Model and Its Supporting System. Journal of Computer Science and Technology, 2002, 17(4): 432～441.

[6] Hansky(China) Ltd. Technology White Papers of Firefly. http://www. hansky. com/cn/resources/articles/index. html. (Hansky. Firefly 技术白皮书. http://www. hansky. com/cn/resources/articles/index.html).

[7] Chen Z Q, Zhong, L H, Zhang L, *et al*. Study of Software Change Management System. Mini-micro system, 2002, 23 (1): 29～31.(陈兆琪, 钟林辉, 张　路等. 软件变化管理系统研究. 小型微型计算机系统, 2002, 23 (1): 29～31).

[8] Li K Q, Guo L F, Mei H, *et al*. An overview of JB (Jade Bird) component l ibrary system JBCL. Technology of Object-Oriented Languages, TOOLS 24, Proceedings, 1997.

[9] Casanova M, Van Der Straeten R, Jonckers V. Supporting evolution in component-based development using component libraries. Software Maintenance and Reengineering, Proceedings, Seventh European Conference, 2003,26～28.

[10] Zhong L H, Xie B, Shao W Z. Supporting Component-Based Software Development by Estending the CDL with Software Configuration Information. Journal of Computer Research and Development, 2002, 39(10):1 361～1 365.(钟林辉, 谢　冰, 邵维忠. 扩充 CDL 支持基于构件的系统组装与演化. 计算机研究与发展, 2002, 39 (10):1 361～1 365).

[11] Yang F Q, Mei H, Li K Q. Software Reuse and Software Component Technology. Acta Electronica Sinica, 1999, 27(2): 68～75.(杨芙清, 梅　宏, 李克勤. 软件复用与软件构件技术. 电子学报, 1999, 27(2): 68～75).

[12] Ren H M, Wang Y F, Qian L Q. Modeling and Implementing a Component System in a Component Library. Mini-micro System, 2002, 23 (9): 1 114～1 117.(任洪敏, 王渊峰, 钱乐秋. 构件库中构件系统的模型和实现. 小型微型计算机系统, 2002, 23(9): 1 114～1 117).