第 32 卷 第 1 期 计 算 机 工 程 2006 年 1 月
Vol.32 № 1 Computer Engineering January 2006
·软件技术与数据库· 文章编号：1000—3428(2006)01—0064—03 文献标识码：A 中图分类号：TP39

可变粒度及面向过程的软件配置管理系统

李 娜，钱乐秋，赵文耘，彭 鑫
（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：软件配置管理(SCM)贯穿于整个软件生命周期，是软件开发过程中质量管理的精髓所在。该文在研究了现有的软件配置管理技术成果的基础上，分析了新的软件开发技术的发展所提出的问题和要求以及目前软件配置管理系统的不足，提出了一个支持可变粒度的、面向过程的软件配置管理系统——FDSCM，并依次详细介绍了该系统在配置项的分类、组织、描述和版本管理方面对可变粒度配置管理的支持，以及在支持迭代开发的过程管理方面的具体设计和实现策略。

关键词：软件配置管理；配置项；可变粒度；软件过程；软件复用；软件构架

Research on a Diverse-grained and Process-oriented Software Configuration Management System

LI Na, QIAN Leqiu, ZHAO Wenyun, PENG Xin
（Software Engineering Lab, Dept. of Computer Science and Engineering, Fudan Univ., Shanghai 200433）

【Abstract】 The goal of SCM is to establish and to maintain the integrity of the software product throughout the development lifecycle, and it results in software quality and productivity improvement. This paper proposes a diverse-grained and process-oriented software configuration management system FDSCM, based on the analysis of the actual state of SCM technology and the challenge arose by advanced software development skills. Then it presents the key solutions, such as how to classify, organize, and describe configuration items to support diverse-grained configuration management, and how to support an evolving development environment and an iterative development cycle.

【Key words】 Software configuration management(SCM); Configuration item; Diverse-grained; Software progress; Software reuse; Software architecture

1 概述

1.1 软件配置管理的定义

软件配置管理：是一种按规则实施的管理软件开发和维护过程及其软件产品的方法。它通过在软件开发过程的特定时刻标识一个软件系统的配置，即一组中间软件产品及描述，系统地控制对配置的更改，并在整个软件生命周期中维护配置的完整性和可跟踪性[1]。

1.2 软件配置管理技术目前的研究成果

软件配置管理技术意味着配置管理能力的自动化和提供完成这些能力的基础设施。目前，配置管理技术既以第三方工具的形式出现，又以开发环境一部分的形式出现。

现有的 5 个典型的软件配置管理模型包括：CheckOut/CheckIn (检出/检入) 模型，组织模型，长事务模型，变更集模型，统一 CM 模型[1]。

15 个 CM 概念构成了 CM 技术的核心，包括配置管理库、分布式组件、环境管理、约定、变更请求、生命周期模型、修改集合、系统模型、子系统、对象池、属性、一致性维护、工作空间、透明视图、事务处理[2]。

1.3 SCM 面临的新问题和 FDSCM 系统的提出

长期以来，软件系统的开发大多是从头开始的，采用传统的开发模型，如瀑布模型等，软件开发经历一系列的分析、设计、编码、测试等开发过程，最终形成产品。相应的配置管理系统管理的对象主要是各种文档、代码等文件粒度的资源以及传统的顺序进行的软件开发过程。已有的 SCM 工具，包括面向过程的 CCC/HARVEST、并发版本系统 CVS、比较全面的 Rational ClearCase 等，已经提供了许多针对传统开发方法的软件配置管理技术、方案。

近年来，软件开发的规模不断增大，对软件质量的要求更加严格。同时，软件体系结构、构架技术，软件复用和构件技术，过程建模技术等软件开发新技术不断深入研究并取得了一定的成果。为软件开发引入了新的运作方式，不仅软件开发过程发生了变化，软件开发中所需要管理的资源也发生了变化。

因此，本文以软件配置管理技术现有的研究成果为基础，针对新技术发展提出的问题和要求，设计了支持可变粒度、面向过程的软件配置管理系统 FDSCM，并对此设计进行了实现。

2 FDSCM 系统中配置项的分类、组织、描述和版本管理策略对可变粒度配置管理的支持

2.1 配置项的分类、组织

软件配置，是指一个软件产品在软件生命周期各个阶段所产生的各种形式(机器可读或人工可读)和各种版本的文档、程序及其数据的集合[3]。

该集合中的每一个元素称为该软件产品软件配置中的一个配置项(Configuration Item) [3]。

基金项目：上海市科委基金资助项目（035115026）；国家“863”计划基金资助项目（2002AA114010）
作者简介：李 娜（1980—），女，硕士生，主研方向：软件工程；钱乐秋、赵文耘，教授、博导；彭 鑫，博士生
收稿日期：2005-01-21 **E-mail：**lois_li@sina.com

2.1.1 实体配置项(EntityCI)、复合配置项(CompositeCI)

软件项目开发中，随着软件开发过程的推进，需要纳入管理的各种工作产品的数量不断上升，各种产品之间的关系也越来越复杂，互相之间依赖性增加。若仅以单个文档或代码文件等作为配置管理的单位，对开发人员来说，配置项都在同一层次上，粒度过小，导致配置项数量巨大，配置项之间的关系难以维护，不仅增加了理解难度、软件维护难度，而且无法保证系统整体上的正确性、一致性。

因而，FDSCM 系统将配置项分为实体配置项、复合配置项两类：

定义 1 实体配置项：是由文件实体的资源封装成的基本配置项。是配置管理的基本单位。可以是任何类型的文件或其组合，例如文档、图片、源代码文件或文件夹。

定义 2 复合配置项：形成复合配置项的过程就是进行配置的过程。用户可以根据需求，将一组相关的配置项以一定的结构组织起来，形成一个更高抽象层次上的、粒度较大的复合配置项，复合配置项又可以与其它配置项组成更上一层的更大粒度的复合配置项。这种递归的组织方式，形成了“部分-整体”的层次结构，逐步提升了配置管理的粒度；同时组成复合配置项的各子配置项在配置过程中需要按照组合约束条件进行一致性、正确性的检查，自动控制了配置中配置项之间的关系和依赖性。

复合配置项作为一个较大粒度的配置项，本身也是一个配置管理的单位，它将组成细节和内部依赖关系进行了封装，降低了开发人员理解的复杂性。对外作为一个整体来处理。

2.1.2 结构配置项(StructureCI)

随着软件复用和构件技术的深入研究，软件在开发过程中可以复用已有的软件制品。可复用的软件制品不仅是文档、代码，还包括软件体系结构，即可以创建一组共用的体系结构模板，并将那些模板作为可复用的设计框架。这样，与产品本身一样，体系结构也应该具有独立的演变属性，可以独立演化。因而，FDSCM 将结构也作为配置管理的对象，对结构演化的各个版本进行跟踪管理。

2.1.3 配置项的分类

综上所述，FDSCM 系统进行版本管理的对象既可以是粒度较小的实体配置项，又可以是粒度较大的复合配置项，此外还支持结构的单独演化，配置项的分类设计如图 1 所示。

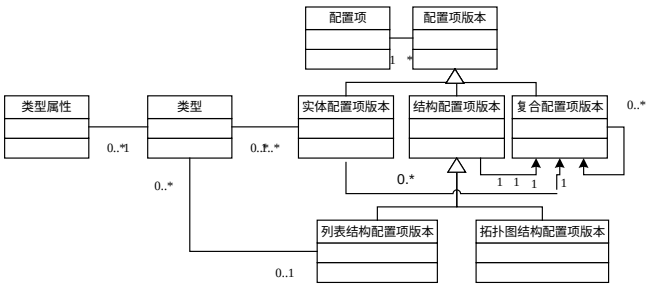

图 1 配置项的分类图 (实体配置项、复合配置项、结构配置项)

2.2 实体配置项的描述和版本管理

在项目开发中某些实体资源被纳入到配置管理中，即被检入到资源库中时，这些资源被封装成实体配置项。相应的描述信息存放在项目配置信息库中，实体的物理存放位置对用户是透明的。作为配置管理的基本对象，实体配置项的描述包括：配置项的基本信息，配置项的类型，配置项的约束条件，配置项的存储信息等。

两层的检出/检入模型，控制配置项版本激增：像大多数配置管理工具一样，FDSCM 系统通过 Check-out 和 Check-in 的方式实现对配置项的版本管理。Check-out 一个配置项，就会在开发者本地的个人工作区中创建该配置项的一个可编辑的版本，可以对该配置项中的内容进行修改；修改后，Check-in 该配置项，就在配置管理库中创建该配置项的一个新的永久的版本，本地的个人工作区中对应的配置项就会变成只读属性，无法修改。

问题在于：一方面，由于在实际开发中，尤其在项目开发阶段，考虑到开发人员可能对资源进行频繁的修改，如果每次所做的修改都在配置库中生成相应配置项的一个新的版本，将会导致配置项版本激增，如图 2 所示(为简化问题，此处未考虑配置项版本的分支、合并)。

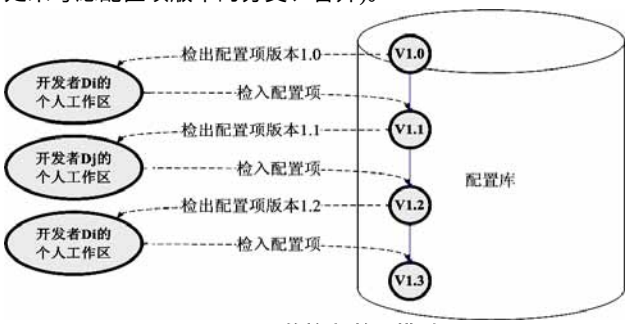

图 2 一层的检出/检入模型

但是，另一方面，有时配置项的一些中间修改版本对开发者个人来说确实又有保存的必要。

因此，为了解决以上二者间的矛盾，采用两层的检出/检入模型，关键要素包括：

(1) 本地配置库：对每个开发者，在本地保存仅对当前开发者有效的小型本地配置库，用来记录开发者在开发、维护过程中需要保存的对其个人有意义的配置项版本信息。

(2) 本地检出/检入：本地对配置项进行修改采用一层检出/检入模型。一次本地的检出，配置项被从本地配置库中检出到本地的个人工作区中进行修改，修改后的配置项从本地的个人工作区中检入到本地配置库中，配置项的本地版本提升 1。本地配置项的变更包括配置项中包含的任何一级的文件的变更，或组成配置项的文件或组织结构发生变化。

(3) 配置项在本地配置库到服务器的项目配置库之间的上传/下载：配置项在本地经过若干版本的演化后，一旦得到一个有意义的版本(例如是若干次调试、修改后的一个较稳定的、可以让其他同组开发者共享的版本)，可以检入到服务器的项目配置库中(称作上传到服务器)，生成相应配置项的一个永久版本——服务器版本，可供其他开发者或他自己检出(称作下载到本地)到本地配置库中后进行进一步修改。这是第二层检出/检入。

两层的检出/检入机制可以控制配置项版本的激增，同时它将各个开发者的开发活动真正独立分开，而且允许多个开发者对同一配置项进行并行开发，如图 3 所示。

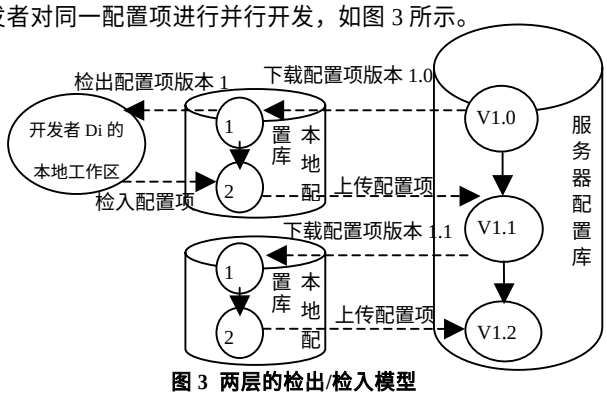

图 3 两层的检出/检入模型

2.3 结构配置项的描述和版本管理

目前，FDSCM 支持的结构配置项有两种：

（1）类型列表：若仅为表示组成复合配置项的各个子配置项的类型约束，则任一个版本结构配置项的描述由一系列配置项的类型列表，及对各个类型由用户根据需要定义各个组成属性的取值约束组成。

（2）拓扑结构：若为表示组装复合构件的子构件之间的连接关系，则任一个版本的结构配置项用于描述构架的拓扑结构，包括组成构架的子构件和连接器，及它们之间的连接关系图。采用构件组装模型的形式化描述语言来进行描述。

不论是描述列表约束还是拓扑结构，结构配置项所描述的任一组成信息或它们之间的顺序、连接关系发生变化，则结构配置项的版本提升。但结构配置项不存在实体，而且创建和产生新版本都是组装和设置约束的过程，相对较快，因而都是在线时进行创建或排他地修改的。

2.4 复合配置项的描述和版本管理

复合配置项某版本的生成就是进行配置的过程，只不过采用复合配置项的概念，将它对外也作为一个配置项统一纳入配置项的管理，使它的粒度相对提升，而且更加抽象。

配置复合配置项是一个选取模板并填充模板组成部分的过程。相应地，复合配置项的描述包括结构配置项版本，各个组成的子配置项版本。复合配置项中的组成子配置项或其版本的替换会导致复合配置项版本变化；结构配置项替换，所有组成子配置项版本要重新选择，也会导致复合配置项版本变化。

3 FDSCM 系统的项目结构组织策略及系统对软件开发过程的支持

3.1 项目结构的组织

FDSCM 中各个项目之间是独立的，每个项目采用递归的分层组织模型，如图 4 所示。特点是：

（1）项目是由若干子系统或构件组成的一个整体。子系统是整个项目开发任务分解所得的某个子部分，它的划分由项目设计者根据具体项目设计定义。构件是一个独立演化、独立进行开发和过程控制的单位，包括分析、设计、编码等各阶段产生的软件资源。构件系统中的构件就是这个含义上的构件。

（2）子系统可以包含其它子系统或构件；构件只能在各版本上由其它构件进行复合，而不能再包含子系统或构件，可认为构件是本次项目开发中对以前开发成果的复用，作为一个整体引入项目的。

（3）每层子系统或构件有自己需求、设计、实现(包括对下层子系统或构件的控制、组装)等过程，各过程由它们各自包含的配置项组成。

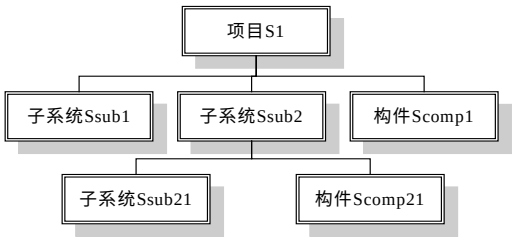

图 4 项目结构

3.2 软件开发过程的定义

可以对软件开发的过程步骤进行定制，用户可以灵活地为不同软件项目选择合适的过程模板或者使用自己定制的过程模板，以选取软件开发过程中的步骤。

（1）过程模板：过程模板定义了软件开发过程中一系列的关键活动，称为过程，并定义了各过程之间的前驱后继关系。过程模板由系统管理员定义，项目建立时选择一个模板，该模板将用于系统中的所有子系统。构件由于可以单独演化，因此构件可以选择自己的过程模板。例如，传统的瀑布模型中的过程包括：需求分析—概要设计—详细设计—编码—测试—发布，并且是顺序推进的。

（2）可变粒度的过程管理：子系统或构件是过程管理的单位，从而过程管理将在各层子系统粒度上进行。

每个配置项有明确的过程属性，属于一个项目中一个特定子系统或构件的一个特定的开发过程。明确了项目的组织结构和过程定义之后，可以看到，对每个配置项的描述又增加了所属的项目结构部分和所属的过程这两个属性，如图 5 所示。

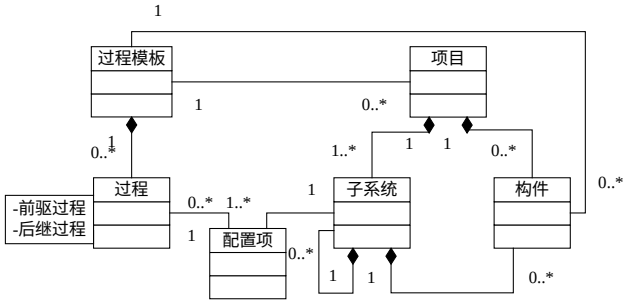

图 5 项目结构和过程管理图

3.3 支持循环迭代式的软件开发周期

过程集成基线：每个子系统的每个过程在开发中都会产生一些资源，这些资源被封装成一些属于该子系统、该过程中的配置项。过程开发结束，选取属于该过程的各配置项的某个特定的版本，组成一个过程集成基线。是一种特殊的复合配置项。

FDSCM 系统支持对反馈和迭代式的软件开发过程的管理。也就是软件生命周期可以循环。因此，创建好的某过程集成基线中的各组成配置项及其版本，可以在新一轮的循环开发中发生变化，或增加、被删除、被替换，从而导致过程集成基线生成新的版本，称为过程集成基线的纵向版本提升。导致版本变化的原因有两种：

（1）过程集成基线版本自主变更，这种情况下引起一轮新的生命周期循环。

（2）由前驱过程的集成基线版本变更激发的本过程集成基线版本变更，这种情况下延续一个已有的生命周期。过程集成基线版本的变化是由该过程的某个前驱过程的某个版本变化引起的。前驱过程版本纵向提升导致当前过程版本纵向提升称为横向变更激发。

4 结语及下一步工作

本文提出了一个可变粒度、面向过程的软件配置管理系统 FDSCM。

该系统通过将配置项分为实体配置项、复合配置项，可以根据用户需要合理组合配置项，提升配置项的粒度和抽象层次并可进行自动化的一致性检查，有力地支持了可变粒度的配置管理；通过引入结构配置项，为配置管理系统扩充了对构架的单独演化功能的支持。这些都为进一步支持软件复用的开发方法提供了扩充接口。

该系统通过提供对项目进行分层结构组织、定义过程模板等功能，支持可变粒度的过程管理；通过提供可循环的软件开发生命周期管理，支持迭代的软件开发过程控制。

目前系统已经完成了设计和原型开发，并将在部分中小型软件开发企业中使用。下一步的工作是对软件复用和构架技术、基于构件的开发和构件组装技术进行深入研究，使得系统可以进一步支持软件复用。

（下转第 150 页）

2.4 总体框架

虚拟设备 dum 是用户与网络接口直接交互的中介，具有极其重要的地位。它的数据组织结构应力求简单清楚，我们是这样安排的：

dum_dir_root

rxFreeQ
rxBusyQ
txFreeQ
txBusyQ

phy_addr_table
…

其中 dum_dir_root 结构记录了环形队列 FreeQ 和 BusyQ 以及地址映射关系表 phy_addr_table 相对于它自身的偏移地址。这样，用户应用层与驱动程序只需共享这一数据结构就能够根据 4 个队列与地址映射关系表 phy_addr_table 进行相互之间的数据传递了。因此，根据前面讲述的，当用户对 dum 虚拟设备进行 mmap 系统调用，以及地址映射关系表时，虚拟设备 dum 将 dum_dir_root 分别传给用户层与驱动程序。最终的基于零拷贝思想面向用户的通信协议总体框参见图 2。

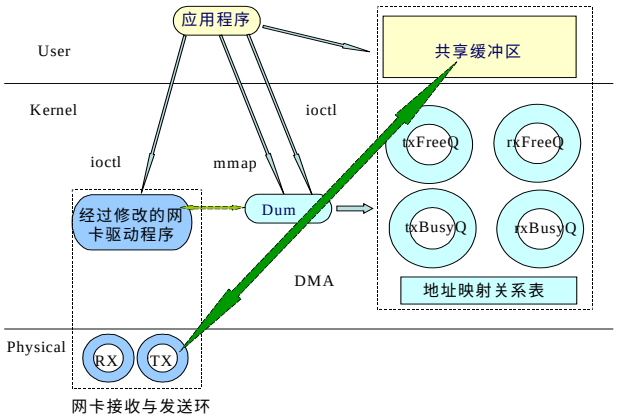

图 2 基于零拷贝思想的用户级通信协议总体框图

3 性能分析

为了测试本文实现的基于零拷贝思想面向用户的通信协议的具体性能，我们在曙光服务器（配置：PIV2.0GHz×2、4GB 内存、Inter pro 1000 网卡）上安装 linux7.3 操作系统，并加载了虚拟设备 dum 以及修改过的专用网卡驱动程序，与专用硬件发包机 SmartBits-6000(B)通过光纤互联，组成测试环境。测试结果如图 3。从图 3 中可以看到，正如前面所阐述的，通过减少数据在操作系统中拷贝次数、简化消息传递途径中的处理环节的方法而减少延迟的基于零拷贝思想面向用户的通信协议，在处理大小为 1 500B 的 IP 数据报文时，最高收包流量为 895Mbps，接近 1Gbps，发挥了网卡性能的 89.5%，与利用传统协议栈的 libpcap 相比，在流量上的性能提高是非常明显的。但是基于本文的设计，实际上将部分传统的协议处理转移到用户层完成，自定义用户接口，因此用户和网络接口卡不能够检验由 dum 虚拟设备进行转换的物理地址的有效性，这还需要在以后进一步的研究。

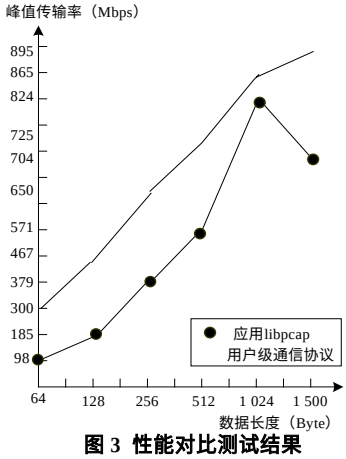

图 3 性能对比测试结果

4 结束语

基于零拷贝思想而设计的面向用户的底层通信协议采用了用户态、内核态之间地址转换的技术，提高了用户访问网络界面的灵活性和效率，降低了消息在传送路径上的延迟，为网络并行计算的机群系统提供了高速度、低消耗的有效保证。对于实现网络高带宽、低延迟数据传输有着极大的意义。

参考文献

1 Chang Sheue ling, David Hung Chang Du, Hsieh J, et al. Enhanced PVM Communications over a High-speed LAN[J]. IEEE Parallel and Distributed Technology, 1995, 3(3)

2 Hsiao K, Chu J, SunSoft Inc. Zero-Copy TCP in Solaris[C].Use Nix Annual Technical Conference. http: //citeseer.ist.psu.edu/chu 96zerocopy.htm/, 1996

3 Eicken T V, Basu A, Buch V, et al. U-Net: A User-level Network Interface for Parallel and Distributed Computing[C]. Proc. of the 15th ACM Symposium on Operation Systems Principles, Copper Mountain, Colorado, 1995-12:3-6

4 Bhoedjang R, R¨uhl T, Bal H E. Design Issues for User-level Network Interface Protocols on Myrinet[D]. Dept. of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands, 1998

（上接第 66 页）

参考文献

1 徐晓春，李高健．软件配置管理[M]．北京：清华大学出版社，2002

2 Hass A M J. 龚　波，黄惠平，王高翔译. Configuration Management Principles and Practice[M]. 北京：清华大学出版社，2003

3 GB/T 12505-90，计算机软件配置管理计划规范[S]. 1990

4 Ohst, Kelter U. A Fine-grained Version and Configuration Model in Analysis and Design[C]. Proceedings of the International Conference on Software Maintenance (ICSM02), 2002

5 梅　宏，张　路，杨芙清. A Component-Based Software Configuration Management Model and Its Supporting System[D]. 北京：北京大学软件工程研究所, 2000