原版风暴·软件工程系列

PEARSON
Addison Wesley

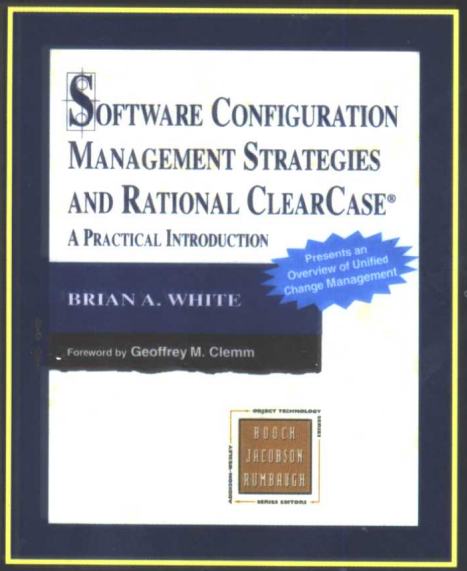
Software Configuration Management Strategies and Rational ClearCase

软件配置管理策略与Rational ClearCase

（影印版）

总览
统一变更管理
（UCM）

[美] Brian A. White 著



Pearson Education
中国电力出版社
www.infopower.com.cn

[illegible]风暴 • 软件工程系列

Software Configuration Management Strategies and Rational ClearCase

软件配置管理策略与 Rational ClearCase

（影印版）

[美] Brian A. White 著

中国电力出版社

Software Configuration Management Strategies and Rational ClearCase
(ISBN 0-201-60478-7)
Brian A. White
Copyright © 2000 Addison Wesley
Original English Language Edition Published by Addison Wesley
All rights reserved.
Reprinting edition published by PEARSON EDUCATION NORTH ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2003.

本书影印版由 Pearson Education 授权中国电力出版社在中国境内（香港、澳门特别行政区和台湾地区除外）独家出版、发行。
未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。
北京市版权局著作合同登记号：图字：01-2003-1009
For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).
仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

图书在版编目（CIP）数据

软件配置管理策略与 Rational ClearCase /（美）怀特著. —影印本. —北京：中国电力出版社，2003
（原版风暴·软件工程系列）
ISBN 7-5083-1400-X
Ⅰ.软... Ⅱ.怀... Ⅲ.软件—配置—管理—英文 Ⅳ.TP31
中国版本图书馆 CIP 数据核字（2003）第 048303 号

责任编辑：乔晶
丛 书 名：原版风暴·软件工程系列
书　　名：软件配置管理策略与Rational ClearCase（影印版）
编　　著：（美）Brian A. White
出 版 者：中国电力出版社
地址：北京市三里河路6号　邮政编码：100044
电话：（010）88515918　传真：（010）88518169
印　　刷：北京地矿印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印　　张**：21
书　　号：ISBN 7-5083-1400-X
版　　次：2003年7月北京第一版
印　　次：2003年7月第一次印刷
定　　价：38.00 元

Foreword

As chief engineer for the Configuration Management business unit of Rational Software, I have the pleasure of focusing on a fascinating problem: how to do software configuration management "right."

My first experience with the rewards and perils of "doing it right" was as an undergraduate in applied mathematics at Harvard University. My freshman mathematics professor turned out to be a 35-year-old graduate student with one goal: solving an elegant little puzzle that had baffled mathematicians for a century, the Four-Color Problem. His reward was that he got up every morning knowing exactly what he was going to do. The peril was that fifteen years later, he was still a graduate student. Happily, any immediate risk of my following that perilous path was eliminated when later that year I had my first encounter with software engineering (back then, we just called it programming). Programming was mathematics with a built-in positive-feedback mechanism. You get the proof (the program) right, and you have not only intellectual gratification, but also something that actually does something interesting and for which companies are eager to pay large sums of money.

But later, in graduate school, the positive-feedback mechanism started to break down. By that time, the software systems I built involved combining large numbers of files from a variety of programmers, some of whom worked even more irregular hours than I did. Instead of composing elegant algorithms, I spent the majority of my time finding out what others had done and letting them know what I planned to do. Not only was this far less intellectually gratifying, but everything took much longer to build, and funding agencies were correspondingly less eager to pay for results that came ever more slowly.

So this would be my Four-Color Problem: discovering the process by which an individual could be a member of a team of programmers but be as productive as when working alone. Attempts to solve the problem by introducing policies and procedures for managing change did make the process more

predictable and decrease certain classes of errors, but at the cost of further decreasing both the intellectual gratification and the productivity of the programming process.

One ray of hope came from the cute little program called "make." No policies or procedures were required. With only file system date stamps and an admittedly cryptic "makefile," the work from an arbitrarily large number of programmers could be compiled as reliably and efficiently as that of a single programmer working alone. The challenge was to find an analogous software tool that would reduce or eliminate the overhead of programming in a team environment. (As a side note, around that time, the Four-Color Problem was finally solved, but only with the aid of a computer program. Apparently, even some mathematical problems are tractable only with the aid of software tools.)

During a decade of studying the problem of software configuration management at Inference Corporation, Sun Microsystems, Hewlett-Packard, and Bellcore, I had the opportunity to identify and implement several key components of automated software configuration management. Then, in November of 1995, I was asked to join Atria Software as project lead for "ratbert," the code name for a project to build a process automation layer above ClearCase.

One of my first activities at Atria was to participate in the design of an "out-of-the-box" process for the new process automation product. Initially assigned to implement this process was a soft-spoken sales engineer named Brian White. Although it seemed a bit strange to have someone in the sales organization be responsible for implementing such a key element of a new product, it soon became apparent that sales engineers at Rational were some of the world's experts on SCM processes. Their job was to go to the most sophisticated software development organizations in the world, learn about how those organizations did software development, and then show them how to use ClearCase to automate that process. We needed the best sales engineer we could get for this project, and Brian was the person we picked.

The initial design meetings for the out-of-the-box process were held off-site at the almost-completed new home of Dave Leblang (the chief architect of ClearCase). There were occasional interruptions intrinsic to the completion phase of a successful entrepeneur's new home: Irish stonemasons building granite walls for the terrace; 40 loads of earth brought in to level out the back gardens. But, gradually, the out-of-the-box process took shape.

Although we rapidly converged on the key elements of the process, two distinct viewpoints emerged. One viewpoint was that every organization was very different, and any process that we developed was just some initial code that would be modified significantly by every installation. The other viewpoint,

此为试读,需要完整PDF请访问: www.ertongbook.com

shared by Brian and me, was that a wide range of organizations could share a common process if that process was designed to be easily configured. This viewpoint was supported by the fact that ClearCase sales engineers tended over time to develop a set of standard scripts that they would use with little modification at each new organization. This viewpoint was also supported by my experience that most process variations are added to address missing underlying tool support; but when full tool support is provided for the key SCM functions, a remarkably similar set of processes become appropriate for a wide range of organizations.

In the most recent release of ClearCase, the common process viewpoint has been realized in the form of the "unified change management" process. This takes the form of both a process and tool support specifically designed to support that process (e.g., new objects, operations, and GUIs). Although tool support for this process will need to evolve, it is my belief that this common process will become the standard way of performing software configuration management. Brian White's book, *Software Configuration Management Strategies and Rational ClearCase®*, provides the foundation for both understanding and adopting this process.

Geoffrey M. Clemm, Ph.D.
Rational Software Corporation

Preface

What This Book Is About

This book is about the engineering discipline of software configuration management (SCM) and how the widely used SCM product, Rational ClearCase®, automates and supports SCM best practices through a model called unified change management (UCM). This book covers basic SCM concepts, typical SCM problems encountered as projects and software systems grow in size and complexity, and how you can apply SCM tools and processes to solve these problems. Advanced SCM topics are also discussed, such as managing large geographically distributed teams and combining the disciplines of SCM and change request management (or defect tracking).

Specifically, this book discusses SCM in terms of a specific SCM tool called ClearCase. Although the discussion is specific to ClearCase, some of the material covered is SCM-tool-neutral. There are very few new books on software configuration management and even fewer that provide strategies for a specific tool. It is in the application of an SCM tool where projects most often run into problems and fail.

ClearCase is a commercially available SCM tool. It is a good choice of tool for this discussion because it provides an open architecture that is used to implement and automate a wide range of SCM solutions. ClearCase is used in many different development environments on many types of applications, such as mission-critical IT, embedded systems, telecommunication systems, financial applications, Web site content, and other commercial and government software systems. Companies in these diverse industries are successfully using ClearCase as the cornerstone of their SCM environment.

This book is not a step-by-step cookbook for using ClearCase, nor does it serve as a substitute for the ClearCase product documentation. You can use the concepts in this book to improve your application of any SCM tool. However, you will get the most out of this book if you are planning to deploy ClearCase or you want to improve the current way you use ClearCase.

On a personal note, this book is a collection of the experience I've gained by working with some incredible people in the SCM field over the last ten years. After reading it, you should have a better understanding of software configuration management, a better idea of the software development problems solved by using SCM tools and techniques, and a clear understanding of how you can use ClearCase to solve these problems and meet your SCM requirements. I sincerely hope you enjoy the book and find it valuable.

What You Need to Know before Reading This Book

The key to your success is understanding SCM, the requirements for your software project, and how to apply an SCM tool to meet a project's requirements. This book will get you started if you are new to software configurement management. However, you will get the most out of this book if you already have some SCM experience and have used basic version control tools before. This book assumes you are familiar with the software development process. It will also be helpful if you have a specific development project in mind while you are reading.

Who You Are and Why You Should Read This Book

This book is not about the nitty-gritty details of writing ClearCase triggers and scripting home-grown integrations with legacy tools; rather it will give you a high-level view of some common SCM scenarios and how ClearCase can be applied. If you are new to SCM or ClearCase, read this book cover to cover. If you have used ClearCase or have a strong foundation in SCM, look through the table of contents and pick chapters and sections that are of particular interest to you.

For a Software Engineer

The biggest thing an SCM tool can do for a software engineer is to stay out of the way. SCM should perform its function, yet be as transparent as possible. The SCM tool and how it is applied should maximize your ability to make changes to the software. Poor tools or poorly designed processes can add unnecessary time and effort to your work. This book can help you identify the areas in your SCM tools and processes to streamline. It discusses some new advances in the SCM area specifically designed for streamlining development. One of

these is the notion of activity-based software configuration management. The idea here is to raise the level of abstraction from files to activities. This makes working with an SCM tool, tracking your changes, and sharing changes with other software engineers more intuitive.

If you're new to SCM, read Chapter 1, What Is Software Configuration Management? For an overview of the objects managed by ClearCase, see chapter 4, A Functional Overview of ClearCase Objects. To gain an understanding of how ClearCase is used on a daily basis from a development perspective, see chapter 8, Development Using the ClearCase UCM Model.

For a Software Project Manager or Technical Leader

As a leader for a software project, you are concerned with deciding what changes to make to a software system and then ensuring that those changes happen. Unplanned changes, made by well-meaning developers, introduce risk into the project schedule and may cause schedule delays and poor product quality. The ability to control and track change is essential to your project's success.

This book should help you gain a solid understanding of SCM, see why you need it, and learn how ClearCase can be used to solve problems you may encounter on projects. Specifically, see chapter 6, Project Management in ClearCase, and chapter 7, Coordinating Multiple Project Teams and Other Scenarios. If you are managing teams that are not all in one location, see chapter 10, Geographically Distributed Development, for a discussion of the issues and strategies involved.

For a Tools Engineer

The role of tools engineer is often overlooked but is essential to success, particularly in large organizations. Your job is to figure out how to apply a given tool to the people, processes, and organization for which you work. This book will give you information about SCM and ClearCase that you can use to determine the best way to apply ClearCase to projects.

For Those Evaluating ClearCase

This book is a good starting point in the evaluation of ClearCase because it presents a number of common software development scenarios as well as more complex scenarios such as geographically distributed development. It discusses the requirements of SCM processes and tools in terms of a set of SCM best

practices and shows how to apply ClearCase to support them. Included are overviews of ClearCase's out-of-the-box process, unified change management, and ClearCase objects.

Use chapter 1, What Is Software Configuration Management?, and chapter 2, Growing into Your SCM Solution, to help determine the SCM tool requirements for your project. Look to the remaining chapters to determine whether ClearCase will meet your needs.

For Experienced ClearCase Users

If you are a long-time ClearCase user, this book is interesting from a general software configuration management perspective and may offer some insights into how to approach SCM solutions on your projects. It also offers some advice if you are being asked to support geographically distributed development teams (see chapter 10, Geographically Distributed Development).

The book contains an overview of ClearCase's out-of-the-box usage model called unified change management, which is a recent addition (see chapter 3, An Overview of the Unified Change Management Model). If you are curious about integrating change request management with ClearCase, then look at chapter 11, Change Request Management and ClearQuest. Look also through the table of contents and pick chapters and sections that are of particular interest to you.

How the Book Is Laid Out

Here is a brief summary of all the chapters.

- Chapter 1, What Is Software Configuration Management, provides a general introduction to software configuration management and the key best practices behind it. It answers the questions: what is software configuration management?, what are SCM tools?, and what is the SCM process?
- Chapter 2, Growing into Your SCM Solution, discusses the growing complexity of software development projects and proposes that as projects grow in complexity so does their need for richer SCM support. It covers the history of SCM tool evolution using five categories of software projects ranging from software developed by a single individual to projects with many geographically distributed project teams.

- Chapter 3, An Overview of the Unified Change Management Model, provides an overview of ClearCase's out-of-the-box usage model, unified change management, which automates and supports a particular SCM process. The material is discussed in terms of the roles and responsibilities of the various team members, such as the architect, project manager, developer, and integrator.
- Chapter 4, A Functional Overview of ClearCase Objects, provides a functional overview of ClearCase objects and concepts. This chapter serves as a bridge between general SCM terminology and ClearCase-specific terminology.
- Chapter 5, Establishing the Initial SCM Environment, provides information on setting up an initial SCM environment. It discusses the basics of ClearCase architecture. The chapter also covers mapping the software architecture to the physical components in the SCM tool and briefly discusses creating the SCM repositories and importing existing software.
- Chapter 6, Project Management in ClearCase, focuses on the role of the project manager with respect to SCM. Particular attention is paid to automation and functionality in ClearCase that specifically supports the project manager. It presents an example of creating a ClearCase project.
- Chapter 7, Coordinating Multiple Project Teams and Other Scenarios, discusses the issues of coordinating parallel work. It also covers the scenarios involving multiple teams cooperating on a common release, development of multiple releases in parallel with multiple teams, coordination of IS/IT–style projects, and coordination of documentation-oriented projects.
- Chapter 8, Development Using the ClearCase UCM Model, provides an introduction to using ClearCase, specifically focusing on the role of the developer. It shows you how to find and join an existing project, how to make changes to files to accomplish an activity, how to deliver the changes associated with the activity, and how to update the development workspace with changes made by other developers on the project.
- Chapter 9, Integration, Build, and Release, focuses on the role of the integrator and discusses approaches to software integration. This chapter briefly covers building, baselining, and how baselines are promoted. It provides an overview of how components are staged in a separate repository that is used for delivery and version control of the "built"

此为试读,需要完整PDF请访问: www.ertongbook.com

deliverable files and directories. It also discusses how software is released by comparing different types of software systems.

- Chapter 10, Geographically Distributed Development, discusses the organizational, communication, and technical challenges that need to be overcome to be successful in distributed development. It looks at three common scenarios of distributed development and the issues associated with each. Finally, this chapter discusses the technology provided by ClearCase MultiSite and how to apply MultiSite to the three scenarios.
- Chapter 11, Change Request Management and ClearQuest, covers the area of change request management (CRM), a subset of which is defect tracking. SCM and CRM are two closely related disciplines, which together form comprehensive change management support. This chapter also discusses a product called Rational ClearQuest and how it works in concert with ClearCase to provide the foundation technology for the unified change management model.

Conventions Used

Commands and Emphasized Text

Command line operations are called out with a different font and prompt, for example:

```
prompt> command -flag1 -flag2
```

Long commands are written on multiple lines for clarity (as shown here), but should be typed on one line, for example:

```
prompt> longcommand longobject-identifier
   -flag1 //machine/pathname
   -flag
```

▶ **Note:** Particular points that need to be emphasized appear in the text in this font with an arrow to alert you.

WARNING: The screened warning box is used to emphasize an issue or concern that might be encountered and should be avoided.

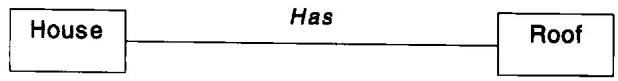
ClearCase Pro Tip

A screened box labeled with the above denotes information that is specifically useful for people who are already using ClearCase. If you have not used ClearCase, you can skip the tips.

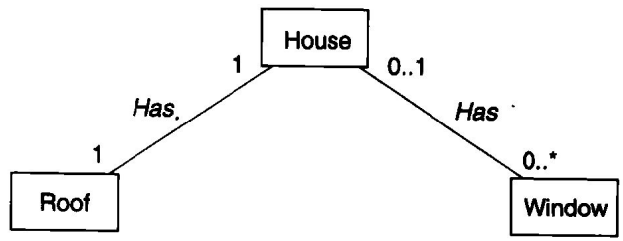
UML Diagram Format

This book includes diagrams that use a graphical modeling language called the unified modeling language, or UML. For more information on UML, see *The Unified Modeling Language User Guide* by Grady Booch, James Rumbaugh, and Ivar Jacobson [Booch 1999].

Here is a description of the small subset of UML used in this book: An object is shown as a box, with text that describes the object. Lines represent associations between the objects, with text that describes the association. For example, "a house has a roof":

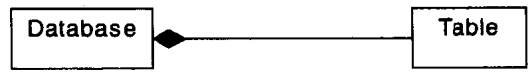


The association can be annotated to provide additional information, such as how many objects can be connected. This is called the "multiplicity" of the association. For example, any given house has only one roof and any given roof can be associated with only one house. Any given house can have many windows or no windows. Any given window can be associated with no house (before it is installed) or one house. These annotations would be represented as shown here:

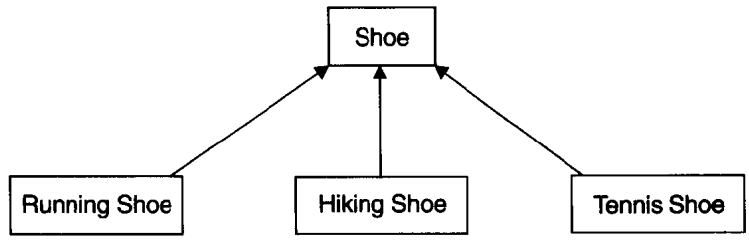


Is it really a house if there are no windows? If not, then you would use "1..*n*" for the windows rather than "0..*n*."

A black diamond is another association annotation that is used to show composition. Composition means one object is composed of another. Important semantics are implied by this type of association. One object "owns" the other. That is, owned objects can be created and removed, but once created they live forever with the owning object. If the owning object is destroyed, its parts are also destroyed. For example, a database has database tables. When the database is destroyed, all the tables are also destroyed. This would be represented in UML as shown here:



Finally, a UML relationship called "generalization" occurs between a general thing and a more specific kind of that thing. For example, the general thing could be a shoe, and specific types of shoes are running shoes, hiking shoes, and tennis shoes. Generalization is represented by an open arrow pointing toward the general object as shown here:



Acknowledgments

I could never have even begun this book without the help of some key people who shaped my career and me. For that I would like to thank Larry Hull, Larry Oslund, Edward Ely, John Leary, and Claudia Dent. After I tossed the idea around for quite a few years, Steve Yost was the catalyst that actually started the project by putting me in touch with someone at Addison-Wesley. I would also like to thank Lorie Stull for her encouragement to take on this challenge.

This book would never have been completed without the help of many others. I would like to thank Kimberly Stamm for her support and encouragement through many long weekends; Claudia Dent for allowing me the flexibility and time I required; the staff of Addison-Wesley—particularly my editor, Debbie Lafferty, and Marilyn Rash; and the production team—Judy Strakalaitis, Hilary Selby Polk, and Doug Leavitt—for walking a new author through the process; Brad Appleton for sharing his depth of knowledge in SCM and for the time spent providing detailed comments throughout many drafts; Arte Kenyon, who was of great assistance in the writing process; and Geoff Clemm, Peter Klenk, Alan Tate, and Nat Mishkin for the finishing touches. I'd also like to extend my thanks to the reviewers: David Bellagio, Ralph Capasso, David Cuka, Elfriede Dustin, Doug Fierro, Susan Goetcheus, Michael Harris, Bill Hasling, Philippe Kruchten, Dean Larsen, Jeff Leyser, Jas Madhur, Linda Okoniewski, Brett Schuchert, Ken Tessier, and all the others who provided invaluable input.

Without ClearCase, this book would not have been written, and without David Leblang ClearCase would never have been built. I would like to sincerely thank the original 10 people who took their fate in their hands and started a company that resulted in the birth of ClearCase: David Leblang, Howard Spilke, Bob Chase, Paul Levine, Dave Jabs, Debbie Minard, Bryan Douros, Gordon McLean, Peter Hack, and Jim Herron. It has been a privilege to have worked directly with most of these incredibly talented individuals who helped set the standard of excellence for SCM tools.

Unified change management would not have been born without the ideas and support of a number of people at Rational Software and at a number of companies using ClearCase. I would like to thank Geoff Clemm, David Leblang, Debbie Minard, and the many ClearCase users I've had the pleasure to work with for formulating the basic ideas behind the UCM model; Peter Klenk for turning those ideas into reality and leading the UCM development effort; Jonathan Aibel, Howard Bernstein, Brian Douros, Hans Heilman, Shirley Hui, Mark Karuzis, Matt Lennon, Nat Mishkin, Brian Morris, Ken Tessier and Mary Utt for the attention to detail and leadership in the design, documentation, implementation, and testing of UCM; Dave Bernstein, Claudia Dent, and Hugh Scandrett for their management leadership; and everyone at Rational Software who is involved with the UCM efforts. Keep up the good work.

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com