Software Configuration Management Patterns: Effective Teamwork, Practical Integration

By Steve Berczuk
with Brad Appleton

Copyright 2002 Addison-Wesley, Boston, MA. All rights reserved
Software Configuration Management Patterns:
Effective Teamwork, Practical Integration

By Steve Berczuk
with Brad Appleton
Copyright 2002 by Addison Wesley ...

All rights reserved.

ISBN 0-000-000000-0

ABCDEFGHIJ-DO-89
Dedication
Table of Contents

Table of Contents iii
Preface ix
  Why I wrote this book x
  Who should read this book xi
  How to read this book. xi
  Origins of this Material xii
  About the Photos xii
Contributor’s Preface xiii
  Why I co-wrote this book with Steve xiii
Acknowledgements xv
1
Introduction 1
  Key Concepts and Terminology 1
  Codeline and Branching Diagrams 6
  Further Reading 7
11
Putting a System Together  11
  Balancing Stability and Progress 12
  The Role of SCM in Agile Software Development 14
  SCM in Context 15
  SCM as a Team Support Discipline 18
  What Software Configuration Management Is 19
  The Role of Tools 22
  The Larger Whole 22
  This Book’s Approach 22
Unresolved Issues 23
Further Reading 23

25
The Software Environment 25
General Principles 25
What Software is About 27
The Development Workspace 30
Architecture 30
The Organization 33
The Big Picture 35
Further Reading 36

37
Patterns 37
About Patterns and Pattern Languages 38
Patterns in Software 39
Configuration Management Patterns 40
Structure of Patterns in this Book 42
The Pattern Language 42
Overview of the Language 44
Unresolved Issues 50
Further Reading 50
51

55
Mainline 55
Simplify your Branching Model 59
Unresolved Issues 61
Further Reading 62

63
Active Development Line 63
Define your goals 66
Unresolved Issues 69
Further Reading 69

71
Private Workspace 71
Isolate Your Work to Control Change 76
Unresolved Issues 80
Further Reading 81

83
Repository 83
One Stop Shopping 86
Unresolved Issues 89
Further Reading 89

91
Private System Build 91
Think Globally by Building Locally 94
Unresolved Issues 98
Further Reading 98

101
Integration Build 101
Do a Centralized Build 104
Unresolved Issues 106
Further Reading 106

107
Third Party Codeline 107
Use the tools you already have 110
Unresolved Issues 113
Further Reading 113

115
Task Level Commit 115
Do One Commit per small-grained task 117
Unresolved Issues 118

119
Codeline Policy 119
Define the Rules of the Road 122
Unresolved Issues 123
Further Reading 123

125
Smoke Test 125
Verify Basic Functionality 126
Unresolved Issues 128
Further Reading 128

129
Unit Test 129
Test The Contract 131
Unresolved Issues 132
Further Reading 132

133
Regression Test 133
Test for Changes 135
Further Reading 136

137
Private Versions 137
A Private History 140

143
Release Line 143
Further Reading 147

149
Release-Prep Code Line 149
Branch instead of Freeze 151
Unresolved Issues 152

153
Task Branch 153
Handling Long Lived Tasks 153
Use Branches for Isolation 156
Referenced Patterns 159
  Named Stable Bases 159
  Daily Build and Smoke Test 159

Bibliography 161

SCM Resources On-line 167

Tool Support for SCM Patterns 171
  VSS - Visual Source Safe 173
  CVS - The Concurrent Version System 175
  Perforce 177
  BitKeeper 179
  AccuRev 181
  ClearCase - base functionality (non-UCM) 183
  ClearCase - Unified Change Management (UCM) 185
  CM Synergy 186
  StarTeam 187
  PVCS Dimensions 189
  PVCS Version Manager 190
  MKS Integrity (Enterprise edition) 191
  Further Reading 192

Photo Credits 1

About the Photos 1
List of Figures 1
Preface

Configuration management is not what I do. I am not a configuration management person. I am not an organizational-anthropology person. However, I discovered early on that understanding organizations, software architecture, and configuration management were essential to doing my job as a software developer. I also find this systems perspective on software engineering interesting. I build software systems, and configuration management is a very important, and often neglected, part of building software systems. In this book I hope that I can show you how to avoid some of the problems that I have encountered so that you can build systems more effectively with your team.

I should probably explain what I mean in making the distinction between SCM people and people who build software systems. The stereotype is that configuration management people are concerned with tools and control. They are conservative, and they prefer slow predictable progress. They are also “the few” as compared with “the many” developers in an organization. Software engineers (so the stereotype goes) are reckless. They want to build things fast, and they are confident that they can code their way out of any situation. These are extreme stereotypes, and in my experience, the good software engineers and the good release/quality assurance/configuration management people have a common goal: they are focused on delivering quality systems with the least amount of wasted effort.

Good configuration management practice is the not the silver bullet to building systems on time, just as patterns, extreme programming, the Unified Process, or anything else that you might hear about. It is however, apart of the toolkit that most people ignore because they fear “process,” often because of bad experiences in the past. (Weigers 2002)

This book describes some common software configuration management practices. The book will be particularly interesting to software developers working in small teams who suspect that they are not using software configuration management as
effectively as they can. The techniques that we describe are not tool specific. Like any set of patterns or best practices, the ease with which you can apply the patterns may depend on whether or not your tool provides explicit support for it.

Why I wrote this book

I started my software development career with a small R&D group that was based in the Boston area. Aside from the many interesting technical problems we had encountered as part of our jobs, we had the added twist of having joint development projects with a group in our parent company’s home base in Rochester, New York. This experience helped me recognize early in my career that software development wasn’t just about good design and good coding practices, but also about coordination among people in the same group, and even teams in different cities. Our group took the lead in setting up the mechanics of how we would share code and other artifacts of the development process. We did the usual things to make working together easier such as meetings, teleconferences and e-mail lists. The way that we set up our (and the remote team’s) software configuration management system to share code played a very large part in making our collaboration easier.

The people who set up the SCM process for our Boston group used techniques that seemed to have been tried throughout their careers. As I move on to other organizations, I was amazed to find how many places were struggling with the same common problems — problems that I knew had good solutions. This was particularly true because I have been with a number of startups that were only one or two years old when I joined. One to two years is often the stage in a startup where you are hiring enough people that coordination and shared vision are difficult goals to attain.

A few years into my career, I discovered patterns. Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides were just finishing the book *Design Patterns* (Gamma et al. 1995), and the Hillside group was organizing the first Pattern Languages of Program (PLoP) conference. There is a lot of power in the idea of patterns since they are about using the right solution at the right time, and also because patterns are interdisciplinary; they are not just about domain or language specific coding techniques, but about how to build software from all perspectives, from the code to the team. I workedshopped a number of papers at the various PLoP conferences that dealt with patterns at the intersection of design, coding, and configuration management (Steve Berczuk 1996b; Stephen P Berczuk 1996a, 1995; Appleton et al. 1998; Cabrera et al. 1999; Steve Berczuk and Appleton 2000).
At one Pattern Languages of Programming (PLoP) conference I met Brad Appleton, who is more of an SCM person than I am. We co-authored a paper about branching patterns (Appleton et al. 1998), just one aspect of SCM. After much encouragement from our peers, I started working on this book.

I hope that this book helps you avoid some common mistakes, either by making you aware of these approaches, or by providing you with documentation you can use to explain methods that you already know about to others in your organization.

**Who should read this book**

I hope that anyone who builds software and uses a configuration management system can learn from this book. The details of the configuration management problem change depending on the types of systems that you are building, the size of the teams, and the environment that you work in. Since it’s probably impossible to write a book that will address everyone’s needs and also keep everyone’s interest, I had to limit what I was talking about. This book will be most valuable to someone who is building software, or managing a software project, in a small to medium size organization where there is not a lot of defined process. If you are in small company, a startup, or in a small project team in a larger organization, you will benefit most from the lessons in this book. Having said that, even if your organization has a very well defined, heavy, process, that seems to be impeding progress, you’ll be able to use the patterns in this book to better focus on the key tasks of SCM.

**How to read this book.**

The introduction explains some basic concepts for software configuration management (SCM) and also the notation that the diagrams use. Chapter 1 introduces the software configuration management concepts that I use in this book. Chapter 2 talks about some of the forces that influence decisions that you make about what sort of SCM environment that you have. Chapter 3 introduces the concept of patterns and the patterns in this book and how they relate to each other. The rest of the book consists of patterns that illustrate problems and solutions to common SCM problems.

Chapters 1 and 2 define the general problems that this book addresses. To understand the how the patterns fit together, you should read chapter 3 to get an overview of the language. After you have read the first 3 chapters, you can browse the patterns
in the rest of the book, starting with an interesting one, and following the ones that relate to your problem. Another approach is to read the patterns in order and form a mental picture of the connections between them.

The references to the other patterns in the book appear in the introductory paragraph for each section, and in the “Unresolved Issues” section at the end of each chapter, using a presentation like this: \textit{ACTIVE DEVELOPMENT LINE (5)}. The number in parentheses is the chapter number that contains the patterns.

Since this is a large field to cover, some of the context and unresolved issues sections don’t refer to other patterns, either in the book, or elsewhere, since they haven’t been documented. In this case you will see a description about what a pattern might cover.

\textit{Origins of this Material}

Much of the material in this book has its origins in papers that were written for various Pattern Languages of Programs conferences by myself, Brad Appleton, Ralph Cabrera, and Robert Orenstein. The patterns here have been greatly revised from the original material, but it’s appropriate to mention these papers here to acknowledge the roles of others to this work: “Streamed Lines: Branching Patterns for Parallel Software Development” (Appleton et al. 1998), “Software Reconstruction: Patterns for Reproducing the Build” (Cabrera et al. 1999), “Configuration Management Patterns” (Steve Berczuk 1996b).

\textit{About the Photos}

The photos that start each chapter are from the Library Of Congress. All of the photos are from the first half of the twentieth century. With the exception of one photo (the photo for \textit{ACTIVE DEVELOPMENT LINE (5)}), they are from the collection: Depression Era to World War II ~ FSA/OWI ~ Photographs ~ 1935-1945: America from the Great Depression to World War II: Photographs from the FSA and OWI, ca. 1935-1945. I chose these pictures because I wanted to provide a visual metaphor for the patterns. Software is an abstract concept, but many of the problems that we solve, particularly the ones about teams, are similar to real world problems. I also have always had an interest in photography and history.

-Steve Berczuk, Arlington Massachusetts, June, 2002.
Contributor’s Preface

Why I co-wrote this book with Steve

I began my software development career in 1987 as a part-time software tools developer to pay for my last year of college. Somehow it “stuck” because I’ve been doing some form of tool development ever since (particularly SCM tools), even when it wasn’t my primary job. I even worked (briefly) for a commercial SCM tool vendor, and part of my job was to stay “current” on the competition. So I amassed as much knowledge as I could about other SCM tools on the market. Even after I changed jobs, I continued my SCM pursuits, and frequented various tool user groups on the Internet.

At one time, I longed to advance the “state of the art” in SCM environments, and kept up with all the latest research. I soon became frustrated with the vast gap between the “state of the art” and the “state of the practice.” I concluded I could do more good by helping advance the state of the practice to better utilize available tools. Not long after that, I discovered software patterns and the patterns community. It was clear these guys were “onto” something important in their combination of analysis and storytelling for disseminating recurring best practices of software design.

At the time, there weren’t many people in the design patterns community that were trying to write-up SCM patterns. SCM is, after all, the “plumbing of software development” to a lot of programmers: everyone acknowledge that they need it, but no one wants to have to dive into it too deeply and get their hands entrenched in it. They just want it to work, and to not have to bother with it all that much.
It didn’t take long for me to “hook up” with Steve Berczuk. We wrote several SCM patterns papers together (with Ralph Cabrera) as part of my ACME project at acme.bradapp.net, and later decided to work on this book. We hope this small but cohesive set of core patterns about integration and teamwork helps the unsuspecting developer-cum-project-lead to survive and thrive in successfully leading and coordinating their team’s collaborative efforts and integrating the results into working software.

Thank you to my wife Maria for her unending love and support (and our daughter Kaeley), and to my parents for their encouragement. Thanks also to my former manager Arbela for her encouragement, support and friendship.

-- Brad Appleton