Preface

Software configuration management is not what I do. I am not a software configuration management person. I am not an organizational anthropology person. However, I discovered early on that understanding organizations, software architecture, and configuration management was 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 I have encountered so that you can build systems more effectively with your team.

I should probably explain what I mean in distinguishing between software configuration management (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, nor are patterns, Extreme Programming (XP), the Unified Process, or anything else that you might hear about. It is, however, a

part 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. As with any set of patterns or best practices, the ease with which you can apply the patterns may depend on whether your tool explicitly supports them.

WHY I WROTE THIS BOOK

I started my software development career with a small R&D group based in the Boston area. Aside from the many interesting technical problems we 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 used the usual things to make working together easier, such as meetings, teleconferences, and e-mail lists. The way 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 moved 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 had been with a number of start-ups that were only one or two years old when I joined. One to two years is often the stage in a start-up 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. Erich 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 Programs (PLoP) conference. There is a lot of power in the idea of patterns because they are about using the right solution at the right time and 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 presented a number of papers in workshops at the various PLoP conferences that dealt with patterns at the intersection of design, coding, and configuration management (Berczuk 1995, 1996a, 1996b; Appleton et al. 1998; Cabrera et al. 1999; Berczuk and Appleton 2000).

At one PLoP conference, I met Brad Appleton, who is more an SCM expert than I am. We coauthored a paper about branching patterns (Appleton et al. 1998), just one aspect of SCM. After much encouragement from our peers, we 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. Because it's probably impossible to write a book that will address everyone's needs and 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 a small company, a start-up, or a small project team in a larger organization, you will benefit most from the lessons in this book. 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 focus better on some of the key tasks of SCM.

HOW TO READ THIS BOOK

The introduction explains some basic concepts of software configuration management and the notation that the diagrams use. Part I provides background information about SCM and patterns. Chapter 1 introduces the software configuration management concepts used in this book. Chapter 2 talks about some of the forces that influence the decisions you make about what sort of SCM environment you have. Chapter 3 introduces the concept of patterns and the patterns in this book and how they relate to each other. Part II consists of patterns that illustrate problems and solutions to common SCM problems. Chapters 1 and 2 also define the general problems that this book addresses. To understand the how patterns fit together, you should read Chapter 3 to get an overview of the language.

After you have read the first three chapters, you can browse the patterns in Part II, starting with one you find interesting and following with 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 chapter and in the Unresolved Issues section at the end of each chapter, using a presentation like this: *Active Development Line (5)*. The number in parentheses is the chapter number that contains the pattern.

Because 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, because they haven't been documented as of this writing. In this case, you will see a description about what a pattern might cover.

ORIGINS OF THIS MATERIAL

Much of the material in this book has its origins in papers written for various Pattern Languages of Programs conferences by me, Brad Appleton, Ralph Cabrera, and Robert Orenstein. The patterns have been greatly revised from the original material, but it's appropriate to mention these papers to

acknowledge the roles of others in 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" (Berczuk 1996b).

ABOUT THE PHOTOS

The photos that start each chapter are from the the Library of Congress. All the photos are from the first half of the twentieth century. With the exception of two photos (the photos for *Active Development Line (5)* and *Private System Build (8)*), 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 we solve, particularly the ones about teams, are similar to real-world problems. Also, I have always had an interest in photography and history.

—Steve Berczuk,
Arlington, Massachusetts, June 2002
steve@berczuk.com
http://www.berczuk.com