Contents

Preface xxi
Introduction xxxiii

PART I THE CORE CM BEST PRACTICES FRAMEWORK .............................1

Chapter 1 Source Code Management .......................................................3

Terminology and Source Code Management ......................... 5
Goals of Source Code Management ........................................... 5
Principles of Source Code Management .................................. 6
1.1 Why Is Source Code Management Important? ................... 6
1.2 Where Do I Start? .............................................................. 7
1.3 Source Code Management Core Concepts ........................ 9
   1.3.1 Creating Baselines and Time Machines ......................... 9
   1.3.2 Reserved Versus Unreserved Checkouts .................... 10
   1.3.3 Sandboxes and Workspaces ....................................... 11
   1.3.4 Variant Management (Branching) ............................. 11
   1.3.5 Copybranches Versus Deltas ..................................... 12
   1.3.6 How to Handle Bugfixes ........................................... 12
   1.3.7 Streams ...................................................................... 14
   1.3.8 Merging ...................................................................... 15
   1.3.9 Changesets ................................................................. 16
1.4 Defect and Requirements Tracking ......................................... 16
1.5 Managing the Globally Distributed Development Team .......... 17
1.6 Tools Selection  ........................................................................ 19
   1.6.1 Open Source Versus Commercial .............................. 21
   1.6.2 Product Maturity and Vendor Commitment ................ 21
   1.6.3 Extensibility and Open API ....................................... 22
   1.6.4 Don’t Overengineer Your Source Code Management .......... 22
1.7 Recognizing the Cost of Quality (and Total Cost of Ownership) .............................................. 23
   1.7.1 Building Your Source Code Management Budget ...... 24
3.9 The Future of Environment Configuration ........................................... 57
Conclusion .......................................................................................... 58

Chapter 4 Change Control ........................................................................ 59
Goals of Change Control ................................................................. 60
Principles of Change Control ............................................................. 60
4.1 Why Is Change Control Important? ............................................. 61
4.2 Where Do I Start? ...................................................................... 61
4.3 The Seven Types of Change Control .......................................... 61
   4.3.1 A Priori ................................................................. 62
   4.3.2 Gatekeeping .......................................................... 62
   4.3.3 Configuration Control ............................................ 62
   4.3.4 Change Advisory Board ....................................... 63
   4.3.5 Emergency Change Control ................................... 64
   4.3.6 Process Engineering .............................................. 64
   4.3.7 Senior Management Oversight ................................ 64
4.4 Creating a Change Control Function .......................................... 65
4.5 Examples of Change Control in Action ...................................... 65
   4.5.1 The 29-Minute Change Control Meeting .................. 66
   4.5.2 Change Control at the Investment Bank ................. 66
   4.5.3 Change Control at the Trading Firm ..................... 67
   4.5.4 Forging Approvals ................................................. 69
4.6 Don’t Forget the Risk ................................................................. 69
4.7 Driving the CM Process Through Change Control ..................... 69
4.8 Entry/Exit Criteria .................................................................. 70
4.9 After-Action Review ................................................................ 71
4.10 Make Sure That You Evaluate Yourself ................................. 71
Conclusion .......................................................................................... 71

Chapter 5 Release Management .......................................................... 73
Goals of Release Management ......................................................... 74
Principles of Release Management .................................................... 74
5.1 Why Is Release Management Important? ................................ 75
5.2 Where Do I Start? .................................................................... 75
5.3 Release Management Concepts and Practices ......................... 76
   5.3.1 Packaging Strategies That Work ............................ 76
   5.3.2 Package Version Identification .............................. 76
## Contents

5.3.3 Sending a Release Map with the Release .......... 77  
5.3.4 What Does Immutable Mean? .............................. 77  
5.4 The Ergonomics of Release Management .................. 77  
5.4.1 Avoiding Human Error ................................. 78  
5.4.2 Understanding the Technology .......................... 78  
5.4.3 Tools from Build Engineering .......................... 79  
5.4.4 Avoiding Human Error ................................. 79  
5.4.5 My Own Three-Step Process ........................... 79  
5.4.6 Too Many Moving Parts ............................... 80  
5.5 Release Management as Coordination .................... 80  
5.5.1 Communicating the Status of a Release ............... 80  
5.5.2 Don’t Forget the Release Calendar ..................... 80  
5.5.3 RM and Configuration Control ......................... 81  
5.6 Requirements Tracking ...................................... 81  
5.7 Taking Release Management to the Next Level ........... 81  
5.7.1 Using Cryptography to Sign Your Code ............... 82  
5.7.2 Operating Systems Support for Release Management 82  
5.7.3 Improving Your RM Process ............................. 82  
5.7.4 Conclusion .................................................. 83  

### Chapter 6 Deployment ........................................... 85  

6.1 Why Is Deployment Important? ....................... 87  
6.2 Where Do I Start? ............................................. 87  
6.3 Practices and Examples ..................................... 87  
6.3.1 Staging Is Key .............................................. 87  
6.3.2 Scripting the Release Process Itself ..................... 89  
6.3.3 Frameworks for Deployment .......................... 89  
6.3.4 What If Bob Makes a Mistake? ........................ 89  
6.3.5 More on the Depot ....................................... 90  
6.3.6 Auditing Your Release .................................... 90  
6.4 Conducting a Configuration Audit ....................... 91  
6.5 Don’t Forget the Smoke Test ............................. 92  
6.6 Little Things Matter a Lot ............................... 92  
6.7 Communications Planning ................................. 92  
6.7.1 Announcing Outages and Completed Deployments .... 93
Contents

6.8 Deployment Should Be Delegated ........................................... 93
6.9 Trust But Verify ................................................................. 93
6.10 Improving the Deployment Process ....................................... 93
Conclusion ..................................................................................... 94

PART II ARCHITECTURE AND HARDWARE CM ......................... 95

Chapter 7 Architecting Your Application for CM ......................... 97

Goals of Architecting Your Application for CM ...................... 98
7.1 Why Is Architecture Important? .............................................. 99
7.2 Where Do I Start? ................................................................. 99
7.3 How CM Facilitates Good Architecture ............................ 99
7.4 What Architects Can Learn From Testers ....................... 99
    7.4.1 Testing as a Service to the Developers ..................... 100
7.5 Configuration Management–Driven Development (CMDD) . 101
7.6 Coping with the Changing Architecture .......................... 101
7.7 Using Source Code Management to Facilitate Architecture ... 102
7.8 Training Is Essential ............................................................ 102
7.9 Source Code Management as a Service ......................... 103
7.10 Build Engineering as a Service ........................................... 103
Conclusion ................................................................................... 103

Chapter 8 Hardware Configuration Management ....................... 105

Goals of Hardware CM ................................................................. 106
8.1 Why Is Hardware CM Important? ........................................ 106
8.2 Where Do I Start? ................................................................. 107
8.3 When You Can’t Version Control a Circuit Chip ............... 107
    8.3.1 A Configuration Item by Any Other Name .......... 107
    8.3.2 Version Control for Design Specifications ........ 108
8.4 Don’t Forget the Interfaces ................................................... 108
8.5 Understanding Dependencies ............................................. 108
8.6 Traceability ........................................................................... 108
8.7 Deploying Changes to the Firmware ................................ 109
8.8 The Future of Hardware CM .............................................. 109
Conclusion ................................................................................... 109
PART III THE PEOPLE SIDE OF CM ........................................ 111

Chapter 9 Rightsizing Your Processes ...................................... 113

  Goals of Rightsizing Your CM Processes .............................. 114
  9.1 Why Is Rightsizing Your Processes Important? ................. 115
  9.2 Where Do I Start? ...................................................... 115
  9.3 Verbose Processes Just Get in the Way ......................... 116
  9.4 SPINs and Promoting the CMM ................................. 117
  9.5 Disappearing Verbose Processes ................................. 117
    9.5.1 Agile Processes Just Work ..................................... 118
    9.5.2 Open Unified Process .......................................... 118
    9.5.3 Getting Lean ..................................................... 119
    9.5.4 An Extremely Brief Description That I Hope
           Motivates You to Take a Closer Look at Lean
           Software Development ........................................ 119
  9.6 The Danger of Having Too Little Process ...................... 120
  9.7 Just-in-Time Process Improvement ............................ 120
  9.8 Don’t Overengineer Your CM .................................... 120
  9.9 Don’t Forget the Technology .................................... 121
  9.10 Testing Your Own Processes ................................. 121
  9.11 Process Consultation .............................................. 122
    9.11.1 Transparency That Is Genuine ............................. 122
  9.12 Create a Structure for Sustainability ....................... 122

Conclusion ........................................................................ 123

Chapter 10 Overcoming Resistance to Change ......................... 125

  Goals of Overcoming Resistance to Change ....................... 126
  10.1 Why Is Overcoming Resistance to Change Important? .... 127
  10.2 Where Do I Start? .................................................. 127
  10.3 Matching Process to Culture ..................................... 127
  10.4 Mixing Psychology and Computer Programming ............ 129
  10.5 Process Improvement from Within .......................... 129
  10.6 Picking Your Battles ............................................... 131
  10.7 Fostering Teamwork .............................................. 131
  10.8 Why Good Developers Oppose Process Improvement .... 132
  10.9 Procedural Justice .................................................. 132
10.10 Input from Everyone .............................................................. 132
10.11 Showing Leadership .............................................................. 133
10.12 Process Improvement People May Be the Problem .......... 133
10.13 Combining Process and Technology Training .................. 134
10.14 Listening to the Rhythm ....................................................... 135
10.15 Processes Need to Be Tested ............................................... 136
10.16 Baby Steps and Process Improvement ............................... 136
10.17 Selling Process Improvement .............................................. 137
10.18 What’s in It for Me? ............................................................. 137
10.19 Process Improvement as a Service .................................... 137
10.20 Guerrilla Tactics for Process Improvement ....................... 138
Conclusion .................................................................................. 139

Chapter 11  Personality and CM: A Psychologist Looks at
the Workplace ............................................................................. 141

Goals of Understanding Personality: What’s in It for Me? ....... 142
11.1 Personality Primer for CM Professionals ......................... 144
11.2 What Do CM Experts Need to Consider in
Terms of Personality? ............................................................... 146
  11.2.1 Communication Styles .................................................... 147
  11.2.2 Do Men and Women Use and Interpret
      Language Differently? ....................................................... 147
  11.2.3 Effective Consultation .................................................... 148
  11.2.4 Verifying the Message ................................................... 148
  11.2.5 Information Processing Preferences ............................. 149
  11.2.6 Birth Order at Work ..................................................... 150
  11.2.7 Firstborns as Leaders .................................................. 150
  11.2.8 The Middle-Born Compromiser ................................. 151
  11.2.9 The Youngest as Initiator ............................................. 151
  11.2.10 The Only Child ......................................................... 151
  11.2.11 Being Yourself .......................................................... 152
11.3 Applying Psychology to the Workplace ............................... 152
  11.3.1 Effective Teamwork Begins at Home ......................... 153
  11.3.2 Volleyball or Effective Collaboration ........................... 153
  11.3.3 Embedding Build Engineers and Testers in
      the Development Team ................................................... 153
  11.3.4 Blackbox Versus Whitebox Versus Graybox ............ 154
11.3.5 Group Dynamics That Can Damage the Organization .................................................. 154
11.3.6 Where CM and QA Fit In .......................................................... 154

11.4 Family Dynamics! ............................................................. 155
  11.4.1 Indecisiveness ........................................................ 155

11.5 Workplace Culture and Personality ....................................... 156
  11.5.1 Personality and Structure ...................................... 156
  11.5.2 We Already Invented All the Good Ideas .......... 157
  11.5.3 Loose Cannons Who Don’t Want to Comply ...... 157
  11.5.4 Enforcing Process, While Still Keeping the Train Moving ........................................ 158
  11.5.5 Formulas for Success ............................................. 158
  11.5.6 Caveats .......................................................... 159

Conclusion .............................................................................. 159

Chapter 12 Learning From Mistakes That I Have Made ......................... 161

  Goals of Learning from Mistakes ................................................. 162
  12.1 Why Is It Important to Learn from Our Mistakes? .......... 162
  12.2 Where Do I Get Started? ..................................................... 162
  12.3 Understanding Our Mistakes .............................................. 163
  12.4 The Mistakes I Have Made ................................................. 163
    12.4.1 Missing the Big Picture ........................................ 163
    12.4.2 Writing Release Automation Can Be Challenging .. 164
    12.4.3 Thinking That a Good Process Will Carry Itself ... 165
    12.4.4 Failing to Gain Consensus ...................................... 165
    12.4.5 Failing to Show Leadership for CM ...................... 165
    12.4.6 Becoming Part of the Problem ............................... 165
    12.4.7 Forgetting to Ask for Help .................................... 166
  12.5 Turning a Mistake into a Lesson Learned ................................ 166
    12.5.1 Clarifying What I Need to Get the Job Done ...... 166
    12.5.2 Getting the Training That I Need ....................... 167
  12.6 Common Mistakes That I Have Seen Others Make ............... 167
    12.6.1 Ivory Tower .......................................................... 167
    12.6.2 Failing to Get Technical and Hands-On ............... 167
    12.6.3 Not Being Honest and Open ................................. 168

Conclusion .............................................................................. 168
PART IV COMPLIANCE, STANDARDS, AND FRAMEWORKS ....... 169

Chapter 13 Establishing IT Controls and Compliance ....................... 171
Goals of Establishing IT Controls and Compliance ......................... 172
13.1 Why Are IT Controls and Compliance Important? ...................... 173
13.2 How Do I Get Started? .......................................................... 173
13.3 Understanding IT Controls and Compliance ............................ 174
  13.3.1 Sarbanes-Oxley Act of 2002 .................................. 174
  13.3.2 Management Assessment of Internal Controls ...... 174
  13.3.3 Committee of Sponsoring Organizations ............... 175
  13.3.4 Cobit as a Framework for IT Controls .................. 176
  13.3.5 What Does It Mean to Attest to And Report
      on the Assessment Made by the Management? ...... 176
  13.3.6 Health Insurance Portability and Accountability
      Act of 1996 .............................................................. 177
  13.3.7 When the GAO Comes Knocking .................................. 177
  13.3.8 Results of the Audit ............................................... 178
  13.3.9 GAO Reports on NARA’s Configuration
      Management Practices ............................................. 179
  13.3.10 ERA Configuration Management Plan ................ 179
  13.3.11 Areas for Improvement ........................................ 180
  13.3.12 Understanding the Results of the Audit ................ 180
  13.3.13 Office of the Comptroller of the Currency .......... 181
13.4 Essential Compliance Requirements .................................... 181
  13.4.1 Providing Traceability of Requirements to
        Releases .............................................................. 182
  13.4.2 Production Separation of Controls ....................... 182
13.5 The Moral Argument for Supporting CM Best Practices .... 182
13.6 Improving Quality and Productivity Through Compliance .... 183
13.7 Conducting a CM Assessment ............................................. 183
  13.7.1 Assessment First Steps ........................................ 184
  13.7.2 Listen First Regardless of How Bad the
        Situation Appears .............................................. 184
Conclusion ................................................................................... 185

Chapter 14 Industry Standards and Frameworks ............................ 187
Goals of Using Industry Standards and Frameworks ...................... 188
14.1 Why Are Standards and Frameworks Important? ...................... 188
### Contents

14.2 How Do I Get Started? ................................................................. 189
14.3 Terminology Required ................................................................. 189
  14.3.1 Configuration Item .............................................................. 189
  14.3.2 Configuration Identification .............................................. 190
  14.3.3 Configuration Control ....................................................... 190
  14.3.4 Interface Control ............................................................... 190
  14.3.5 Configuration Status Accounting ...................................... 191
  14.3.6 Configuration Audit .......................................................... 191
  14.3.7 Subcontractor/Vendor Control .......................................... 192
  14.3.8 Conformance Versus Noncompliance ............................... 192

14.4 Applying These Terms to the Standards and Frameworks .. 193
14.5 Industry Standards .................................................................... 193
  14.5.1 IEEE 828—Standard for Software Configuration
    Management Plans ...................................................................... 193
  14.5.2 ISO 10007—Quality Management Systems—
    Guidelines for Configuration Management ............................. 195
  14.5.3 ANSI/ITAA EIA-649-A—National Consensus
    Standard for Configuration Management .............................. 196
  14.5.4 ISO/IEC/IEEE 12207 and 15288 ........................................ 196

14.6 Industry Frameworks ................................................................. 196
  14.6.1 ISACA Cobit ................................................................. 197
  14.6.2 CMM/CMMI ................................................................. 207
  14.6.3 itSMF’s ITIL Framework .................................................. 208
  14.6.4 SWEBOK ....................................................................... 214
  14.6.5 Open Unified Process (OpenUP) ...................................... 215
  14.6.6 Agile/SCRUM ............................................................... 216

Conclusion ....................................................................................... 217

Index .................................................................................................. 219
This page intentionally left blank
Preface

Configuration management (CM) plays a critical role in any technology development effort. I have been involved with implementing and supporting CM for more than 25 years, and much of what I am about to discuss comes directly from my own personal experience. I have implemented and supported each of these CM practices, often with the agreement that I could be woken in the middle of the night if my processes/automation did not work as expected. As an instructor, I have taught industry-strength CM tools to 900+ technology professionals (again with the offer that they got my home phone number upon successfully completing my class). My colleagues and students have consistently indicated that my passion and love for this discipline has always been abundantly clear. It is my view that configuration management consists of six functional areas:

1. Source code management
2. Build engineering
3. Environment configuration
4. Change control
5. Release engineering
6. Deployment

I have searched for, but never found, any single book (or even a series of books) that covered all of these functional areas. Most CM books are either too narrowly focused on one key area (such as building code with Ant) or so “ivory tower” that they did not give me enough information on how to really implement these functions in a practical real-world environment. It’s nice to point out the need to “maintain control of all configuration items,” but unless you tell me exactly how to do that in a practical and realistic way, the advice is not truly usable. It is my intent both to cast a wide net on the CM practices that you need to understand and to provide enough detail so that you know not only what each CM function entails, but, just as important, how to implement each of the CM functions. I expect that my readers will hold me to that commitment. The URL of our supporting website is http://cmbestpractices.com.
The Traditional Definition of Configuration Management

Configuration management, or in this context, software configuration management (SCM), has a traditional definition consisting of four specific functions:

1. Configuration identification
2. Change control
3. Status accounting
4. Configuration audit

These functions have long been described in industry standards and frameworks and obviously viewed as essential to any valid configuration management effort. Although I agree completely that these functions are correct and essential, I find their terminology to be difficult for many technology professionals to understand and appreciate. In this book, I discuss the traditional CM functions, and I suggest a framework for understanding and implementing configuration management in a way that I believe reflects current industry practices. Specifically, I show the relationship between the four classic functions and the six functions of source code management, build engineering, environment configuration, change control, release engineering, and deployment, which I believe more closely reflects the way that CM is actually done on a day-to-day basis. This is an important focus of my efforts to make configuration management best practices more approachable and practical for technology professionals to enjoy as part of their own process-improvement efforts.

Terminology and CM

Configuration management, like many other disciplines, suffers from the use of confusing terminology. I am not going to solve that problem in this book, but I do endeavor to at least not make the situation worse. The acronym SCM has been used to refer to both source code management and, more recently, software configuration management. One of my most knowledgeable colleagues has prevailed upon me to not make the situation worse, so I use the SCM acronym only to refer to the broader software configuration management, which is a specialization of configuration management (as opposed to hardware configuration.
management discussed in Chapter 8, “Hardware Configuration Management”). Similarly, the acronym CI is used to refer to both configuration items and continuous integration. CM terminology can be quite confusing. I can’t do much about the confusion caused by this dual use of CI as an acronym, as it is pervasive, but I do what I can to be as clear as possible. There are similar challenges with regard to the terms configuration control and release management. I do my best to present a clear explanation of these terms and, more importantly, explain how to implement these practices in a real-world setting. Once again, I hope that you will join me online if you want to discuss the use of these terms as well as their evolution. This is an exciting time for configuration management because many technology professionals are recognizing that CM impacts everything from IT service management (ITSM) to the entire Agile ALM. Whenever possible, I endeavor to use the definitions in the IEEE’s SEVOCAB: Software and Systems Engineering Vocabulary, which, at the time of this writing, can be found at www.computer.org/sevocab.

Why I Love CM

I love CM because it is a creative and exciting endeavor that can significantly add value by improving quality and productivity in any technology project. Not only do I discuss what I have learned, but I also relate the combined experience of thousands of CM experts who have kindly shared their own expertise and best practices with me over the years that I have been engaged in this work. I owe each of these fine colleagues a debt of gratitude for all that they have shared with me. I have written and published many articles on configuration management and thoroughly enjoyed the feedback that I have received (especially when those supplying it disagreed with me and offered other practical approaches to solving thorny CM-related problems). I anticipate that this book will also generate considerable interaction with my colleagues, especially through the supporting http://cmbestpractices.com website that I have created. Please visit this website for up-to-date information on the topics that we discuss in this book as well as to give me feedback about your own experiences with implementing configuration management.

Why I Wrote This Book

I wrote this book to share my expertise and experience with implementing all aspects of CM in realistic business, engineering, and government environments.
I hope that you find this information to be practical, comprehensive, and helpful in implementing CM in a variety of real-world situations.

Some topics in CM are evolving so quickly that writing a book on them would be a daunting task. For example, as I write this Preface, my “day job” is to implement IBM’s latest Application Lifecycle Management (ALM) solution, which includes a brand new source code management and automated workflow solution. Therefore, for this book, I discuss how to select a CM tool in only general terms, but restrict tool-specific comments to my supporting website (http://cmbestpractices.com/tools) so that the information can be kept current and accurate. I also hope that you hold me accountable for the accuracy of every word that I write because I have very strong personal views that CM is essential on a moral, ethical, and theological basis. Although CM is not my “religion,” doing honest and high-quality work is certainly part of my religious belief system. I also view spreading CM best practices as being a model for good corporate citizenship. I have been very active in the virtual community that develops and supports configuration management as well as other aspects of application development. On any given day, you can see technology professionals providing each other with substantial assistance without regard for whether they work for competing organizations. The community is truly culturally diverse, multilingual, and universal in its respect for and acceptance of others. I am proud to be part of this work and wish my efforts to promote CM best practices to be part of a wider movement to promote effective IT controls, responsible business leadership, and good corporate citizenship resulting in greater services and value for everyone who shares this increasingly tiny world that we live in. To say this in another way, I believe that every government agency, financial services firm (including banks, hedge funds, and insurance firms), along with firms that are in the medical, pharmaceutical, and defense (and every other) industry should be required to implement proper IT controls to protect the public who rely on their services as well as shareholder value. I wrote this book, in part, to help transition this effort from being a burden to instead being a journey in improving productivity and quality. It is my belief that implementing IT controls, including CM best practices, in a pragmatic way should result in higher profitability for the members of the firms, their shareholders, and the public which depends upon their services.

Classroom Materials

Students and college professors are also welcome to contact us with regard to supporting materials (such as lecture slides, course curriculum, etc.) for the
classroom. Where feasible, it is our intent to offer to visit and lecture in educational settings that adopt this book for classroom instruction purposes. Please contact Leslie Sachs who will coordinate these efforts.

Who Should Read This Book

Technology professionals including development managers, system architects, developers, systems engineers, hardware engineers, quality assurance, quality engineering, operations engineers, and technology project managers will all benefit from the information in this book. CTOs, IT auditors, and corporate managers will especially enjoy the sections on establishing IT controls and compliance. Whether you are an Agile enthusiast or working with a classic waterfall lifecycle, this book will help you get your job done better. CM is all about good corporate citizenship. The news media love to report instances of corporate greed and incompetence among those who have a responsibility for providing and maintaining technology for the public good. CM best practices help ensure that the global economy runs smoothly, ATMs work correctly, air traffic control systems remain online, and so on. If you want your technology development efforts to be more efficient and to yield higher-quality products, this book is for you.

How to Read This Book

You should at least skim the Introduction because it will give you an overview of the CM functions and their overall linkages. You should also feel free to skip to the area that you need help with next. I have endeavored to write each chapter so that it can be read and used separately. In practice, this has often been how I implemented CM. For example, I have often skipped directly to solving the most urgent problems (as indicated by the customer) without being rigid about the order of implementing CM functions. That said, there are some dependencies, and I do my best to describe them, too.

How This Book Is Organized

This book is organized into 14 chapters divided into four parts. Part I consists of six chapters covering source code management, build engineering, change
control, environment configuration, release engineering, and deployment. Part II covers architecture and hardware CM, and Part III covers the essential people issues that you need to know to effectively implement CM best practices. Part IV covers compliance and the standards (such as IEEE, ISO, EIA) and frameworks (such as ITIL, Cobit, CMMI) needed to establish effective IT controls. What follows in the next section is a short description of each chapter.

**Part I: The Core CM Best Practices Framework**

Six chapters make up the core CM best practices framework.

*Chapter 1: Source Code Management*

Source code management is an essential starting point for any configuration management function. In this chapter, we discuss the requirements for an effective source code management effort and some of the core concepts. In source code management, you make sure that you know where all the artifacts needed by your application are located and that they are all properly identified and can be managed effectively. If we were baking a cake, then source code management would help you ensure that you have all the correct ingredients on hand and in the proper amount.

*Chapter 2: Build Engineering*

Build engineering includes the compilation of all the configuration items that go into a release. Your build engineering practices need to be efficient, reliable, and repeatable. Build engineering also includes procedures for building in the essential version IDs that are required for configuration identification. Build engineering involves mixing the batter and baking the cake itself.

*Chapter 3: Environment Configuration*

Environment configuration involves handling the compile and runtime changes necessary for the promotion of code from development to QA to production. It also includes the configuration and management of the requirements. Environment configuration ensures that you have the shelf ready to show off the great cake that you baked.

*Chapter 4: Change Control*

There are seven functions in change control: evaluating requests for change, gatekeeping (such as promotion), configuration control, emergency change control, process changes, advising on the downstream impact of a potential change, and senior management oversight of change control. Change control decides
when the cake is baked and ready to be taken from the oven and sent to the happy person who will enjoy the cake.

**Chapter 5: Release Management**

Release management involves packaging the configuration items into components that can be reliably promoted and deployed as needed. Release management is effectively putting your cake into the nice box with the open window so that others can see and appreciate the fine work that you have done.

**Chapter 6: Deployment**

Deployment should be a narrowly defined function of promoting the prepackaged release to QA or production as needed. This is effectively putting your cake on the truck to be delivered to your consumers. (Make sure that you get my home address correct for delivery.)

This completes the first part of the book, covering what I view as being the essential core CM competencies necessary for any CM function. I am really getting hungry now, so I have to stop using a cake as a metaphor for CM. The rest of the chapters make up the eight supporting functions that are also important for the implementation of an effective CM effort.

**Part II: Architecture and Hardware CM**

Architecture and hardware also are candidates for CM.

**Chapter 7: Architecting Your Application for CM**

This is an often-overlooked aspect of configuration management and involves recognizing the interrelationship between application architecture and configuration management. The essential nature of CM is the same whether you are implementing it on a mainframe or your favorite handheld device. But the actual procedures will vary significantly based on the architecture of your application. So, implementing CM on a WINTEL platform may be very different from on a UNIX/Linux platform using Java SOA or C++. This chapter is about understanding that relationship. This chapter is also about how CM helps implement excellent architecture. CM best practices help your team to develop excellent application and systems architecture.

**Chapter 8: Hardware Configuration Management**

I need to write an entire book on hardware configuration management. There just isn't enough recognition of its value and importance in the CM field. I have
been frequently asked to write about hardware CM. This chapter begins what I am sure will be a longer journey.

**Part III: The People Side of CM**

You can’t afford to ignore the people side of any business or organizational endeavor. CM is no different. I have been involved and observed many successful efforts to implement CM best practices. In the situations where the results were less than acceptable or even truly a failure, it was almost always due to people issues. This chapter gives you very practical advice from real world experiences on how to deal with the people side of CM. This is a very important part of the book for your success.

**Chapter 9: Rightsizing Your Processes**

My whole career has been focused on implementing process improvement. I have learned that too much process is just as bad as not enough. This chapter is about finding the right balance and implementing *just enough* process to get the job done.

**Chapter 10: Overcoming Resistance to Change**

Having a great process does not help anyone if you can’t get your team to accept the process and actually start working in a new and better way. This chapter is about overcoming resistance to change and getting the team to accept and enjoy the new way of doing things.

**Chapter 11: Personality and CM: A Psychologist Looks at the Workplace**

Leslie Sachs takes the lead in this chapter as she describes the essential people skills that you need to be effective in implementing CM best practices. I get scared when I read Leslie’s work because she seems to always be eavesdropping on my conversations. Read this chapter if working with people is important to you.

**Chapter 12: Learning From Mistakes That I Have Made**

I have made lots of mistakes in my career. I have achieved a lot, yet I have also failed to achieve as much as I had hoped. But I have learned a lot from my own mistakes, and this chapter is my effort to share some of my personal improvement efforts to learn from my own mistakes and shortcomings. This chapter could have been its own book or perhaps the size of a small encyclopedia.
Part IV: Compliance, Standards, and Frameworks

The book ends with the issues involved in establishing IT controls, complying with regulations, and the use of industry standards and frameworks. Second only to the people side of CM, understanding industry standards and frameworks is one of the most powerful capabilities that you need to master to successfully implement CM best practices. This information will also help you overcome resistance to change because you will rightly be able to explain what thousands (or perhaps hundreds of thousands) of other technology professionals have reviewed, debated and determined to be the official accepted industry best practices.

Chapter 13: Establishing IT Controls and Compliance

Establishing IT controls and compliance is one of my own favorite topics. I like to focus on using these efforts to improve quality and productivity while you are also getting ready to pass your audit. IT controls and compliance is a really critical topic for many organizations, and I expect that if you need to meet industry regulations, you will find this information to be extremely valuable.

Chapter 14: Industry Standards and Frameworks

I strongly advocate the use of industry standards and frameworks, but I also believe that much of what has been previously written is difficult to understand and even more difficult to implement. I believe that those of us involved with creating industry standards and frameworks need to write more practical material on how to actually implement standards and frameworks in a realistic and pragmatic way. My focus, in this chapter, is on describing my own personal journey with implementing process improvement using the guidance described in standards and frameworks along with the essential skills of tailoring, harmonization, and operationalizing the published guidance. This might be the most important chapter in the book, and I hope that you will give me your feedback on your efforts to embrace and implement industry standards and frameworks.

Overall, I think that you want to focus on the first part of this book to understand the core CM best practices and then read the remaining chapters of this book in whatever order you choose to cover the topics that you have an immediate need for implementing within your organization.
Introduction

In this Introduction, I briefly introduce configuration management (CM) and some basic information on how you might approach implementing CM best practices. It is common for organizations to focus on implementing only a very narrow functional area to address a specific goal or problem. In practice, this might be a perfectly fine thing to do, but it is also important to understand how each functional area of CM impacts the other. It has been my personal experience that CM consists of six functional areas, which I will describe below and throughout this book. Implementing good CM is not easy and requires a considerable amount of hard work. This introduction will help us start our journey.

Configuration Management Consists of Six Functional Areas

The six core functional areas of CM are as follows:

1. Source code management
2. Build engineering
3. Environment configuration
4. Change control
5. Release engineering
6. Deployment

Source code management involves the control of every piece of computer code, including source, configuration files, binaries, and all compile and runtime dependencies. We usually refer to all these artifacts as configuration items (CIs).\(^1\)

The main goal of source code management is to effectively safeguard all the project resources. I always called this locking down the code. Source code

---

\(^1\) Please don’t be confused by the fact that we will refer to continuous integration (CI) with the same acronym.
management also involves creating a permanent record of specific milestones in the development process. This is known as baselining your code, and it is a critical CM function. Source code management also involves creating code variants to successfully manage parallel development, bug fixes, and globally distributed development. We discuss how to assess your source code management requirements and plan interventions to improve your source code management practices. We also look at how source code management is often overengineered, resulting in unnecessary complexity and automation that does not work reliably.

Build engineering involves the selection of a specific variant in the code (e.g., baseline) to reliably compile, link, and package code components. Build engineering adds value by providing a repeatable process and the management of (often complex) compile dependencies. We discuss how to implement effective build engineering to help improve your team’s development process. We also discuss the value of continuous integration (CI) versus the (usually) less-rigorous nightly build.

Environment configuration involves managing the compile and runtime dependencies that can often change as code is promoted from development to test to production. Environment configuration also involves managing the environments themselves often designated as development, test, integration and production.

There are different types of change control. The most commonly implemented change control practice is essentially a “gatekeeping” function that prevents unauthorized releases from being promoted into production (or QA for that matter). There is also a priori change control, whereby intended changes, to the code, are reviewed (before they are made) and permission granted (or denied) to make the proposed changes. We discuss when a priori change control is commonly used and when it is instead left to the project or development manager as an implicit task. We also discuss the other types of change control that are commonly seen in organizations. In all, I define seven different types of change control. I also describe how they are commonly used in practice.

Release engineering involves the packaging and identification of all the components built in the build engineering function. This is somewhat different in a corporate IT function versus a software vendor. We initially focus on corporate release management in a corporate IT environment, and then discuss how this differs slightly for a software vendor (e.g., deploying packaged releases to customers). Deployment involves the staging and promotion of packaged releases and, in an IT organization, is usually performed by the operations team. Deployment also involves the monitoring of the production (and QA) environments to confirm that there are no unauthorized changes. Deployment for a software vendor usually refers to delivering the packaged release to a customer along with the requirement to manage updates and patches as needed.
Configuration Management Consists of Six Functional Areas

All of these functions are part of a comprehensive discipline that is known as configuration management (CM). Software configuration management is a specialization of CM. Equally important and frequently overlooked is hardware CM, which we discuss in Chapter 8, “Hardware Configuration Management.”

Understanding the Linkages

The six functional areas of configuration management impact each other in many ways. Build engineering is almost impossible to do well without effective source code management practices. Release management just won’t happen if your releases are not built correctly, especially in terms of identifying all configuration items, as we describe in Chapter 2, “Build Engineering.” Environment configuration impacts build engineering, release management, and deployment. Of course, deployment is almost impossible if the releases are not packaged correctly. All of these functional areas are impacted by change control best practices. For example, an effective change control board (CCB) will review the CM plan and release management automation before giving permission for the release to be approved. We discuss change control best practices in Chapter 4, “Change Control,” including after-action reviews to ascertain whether mistakes could be avoided by improving any of these configuration management best practices.

The Traditional View of Configuration Management

My colleagues rightly remind me that configuration management is defined as follows:

- Configuration identification
- Change control
- Status accounting
- Configuration audit

They are absolutely correct, and I am not changing the substance of configuration management, but I believe that the terminology used in traditional CM is less than clear and, in this book, I seek to make the terminology that describes configuration management compelling. Generally, I jab back by challenging them to give me a clear and sensible definition for status accounting. In my opinion, the terms configuration identification and configuration audits are not much more intuitive. On the other hand, most developers have a basic idea
of what’s involved with source code management, build engineering, and release management. Let’s bridge the gap with the traditional terminology and then dive deeper into CM.

*Configuration identification* refers to providing a specific and unique identity to each artifact for the purposes of tracking configuration items (e.g., source code, binaries, documents, config files). I have an entire chapter on change control, which I define as being composed of seven functions. *Status accounting* (my least favorite term) refers to tracking the status of a configuration item throughout its lifecycle. *Configuration audits* refer to being able to inspect and identify the exact version of any configuration item. In my opinion, CM experts need to make this terminology easier to understand and use on a day-to-day basis.

For example, configuration identification is actually accomplished by naming the components, streams, and subdirectories (folders) in your source code management tool in a logical and intuitive way. Build engineering best practices enable you to embed version IDs in binary configuration items (it also facilitates configuration audits), and many build tools, such as Maven, help you to organize your code in a logical and sensible way. Release management also involves configuration identification in that you must name your release packages in a clear and consistent way.

Status accounting involves tracking the status of a configuration item throughout its lifecycle. In practice, many source code management solutions are integrated with requirements and defect tracking systems (if not already built in) so that you can easily trace the evolution of a component from its requirement (or perhaps defect record) all the way through to its deployment. Configuration audit mean that you know exactly which version of the code is running in production (or QA). Unfortunately, many technology professionals cannot identify the exact version of a binary configuration item after the code leaves the source code management tool. In my world, you need to be able to tell me the exact version of the code that is running in production (or QA) and be able to retrieve the exact version of the source code used to build it so that you can also create a sandbox (in a source code management tool) and make a small change to the code—without any chance of the code regressing due to the wrong version of a header file or other dependency. If you can’t do that today, you have come to the right place!

The first six chapters of the book make up Part I, “The Core CM Best Practices Framework,” which describes the core functions in configuration management. I describe how the six core functions relate to the traditional view of CM. I also cover a number of other essential topics in Chapters 7 through 14, which are presented in Parts II through IV. Here is a description of these sections.

The second part of the book, Part II, “Architecture and Hardware CM,” deals with understanding the impact of architecture on CM best practices and
The Goals of Good CM

I believe that there are three basic goals that any CM effort must accomplish. The first is that all code that has been deployed to production (or QA) must be easily identifiable. In CM terminology, we call this a configuration audit. That means that you can easily confirm that you know the exact versions of all configuration items in production (with absolute certainty). The second goal is that you can retrieve the exact version of all source code (and other configuration items) used to create that release (without having to resort to “heroic” efforts). Finally, you must be able to create a workspace (often called a sandbox) to make a small “bugfix” without any chance of the code regressing due to the wrong version of a header file (or other dependency). If you can’t do these three things, your CM practices need some improvement. The good news is that we describe exactly how to accomplish these goals in practical and realistic terms.
Index

A
a priori change control, 58, 62
active listening, 148-149
after-action reviews, 71
Agile, 118, 216-217
ALM (application lifecycle management), 26
ANSI/ITAA EIA-649-A standard, 196
Ant, 38
  for complex builds, 39-40
  Maven versus, 39
Appleton, Brad, 121
application architecture. See architecture
application lifecycle management (ALM), 26
applying psychology. See psychology
approval process in change control, 69
architecture, CM (configuration management) and, 94-99
build engineering in, 103
changes to architecture, 101-102
CMDD (configuration management-driven development), 101
goals of, 98
importance of, 99
source code management in, 102-103
starting point for, 99
testing, role of, 99-101
training, 102-103
assessments, configuration management, 183-185. See also audits
asset management in ITIL framework, 210
audits. See also assessments; configuration audits
FDIC audit, 177-179
NARA audit, 179-181

B
bad builds, 31
balancing risk, 158
baselines in source code management, 8-10
Berczuk, Steve, 121
best practices. See also compliance
  in build engineering, 47
  in deployment, 87-90
  in environment configuration, 57
  moral argument for, 182-183
birth order roles in personality, 150-152
  firstborns as leaders, 150-151
  middle-borns as compromisers, 151
  only children, 151-152
  self-expression, 152
  youngests as initiators, 151
blackbox testing, 154
blindness example (process improvement), 116
“Bob method” for training, 24-25
branching in source code management, 11-12
bugfixes, 12-13
  copybraches versus deltas, 12-13
branding executables, 32
breaking rules in workplace culture, 157-158
budget for source code management, 23-24
bugfixes in source code management, 12-13
build engineering, xxxiv, 27-30
  in architecture development, 103
  best practices, 47
  build process improvements, 42-44
  compile dependency management, 33-34
  continuous integration (CI) versus nightly builds, 47-48
cost of quality, 42
ethical issues, 36-37
future of, 48
goals of, 30
importance of, 31
independent builds, 34-35
organizational structure, 37-38
overengineering, 35-36
as part of development team, 153-154
principles of, 30-31
release management and, 79
responsibilities in, 32
role of build engineer, 44-46
starting point for, 32
technology architecture, importance of, 46-47

tool selection, 38-42

version IDs, 32-33
build frameworks, 41

gatekeeping, 62
goals of, 60
hierarchy of, 67
importance of, 61
in ITIL framework, 209
principles of, 60-61
process engineering, 64
risk in, 69
senior management oversight of, 64-65
specialized change control, 67
starting point for, 61
system integration, 70
time management of, 66
change control board (CCB), 61. See also
change control
changesets in source code management, 16
checkouts, reserved versus unreserved, 10-11
CI (continuous integration), 40, 47-48, 216-217
CI servers, 40
CIs (configuration items), xxxiii, 6, 32
defined, 189-190
in hardware configuration management, 107-108
release maps of, 77
CM (configuration management)
architecture and, 94-99
build engineering in, 103
changes to architecture, 101-102
CMDD (configuration management-driven development), 101
goals of, 98
importance of, 99
source code management in, 102-103
starting point for, 99
testing, role of, 99-101
training, 102-103
assessments, 183-185
change control as process driver, 69-70
core functional areas, xxxiv
defined, xxxiv-xxxvi
goals of, xxxvii
hardware configuration management.
See hardware configuration management
personality and. See personality
terminology, xxxiv-xxxvi
CMDB (configuration management database)
centralized environment CMDBs, 55-56
in ITIL framework, 211-213
CMDD (configuration management-driven
development), 101
CMM (Capability Maturity Model), 116-
117, 207-208
CMMI (Capability Maturity Model Integrated),
116-117, 128, 185, 207-208
CMS (configuration management system)
in ITIL framework, 210
Cobit framework, 176-177, 197-207
change control, 198-203
emergency change control, 199
metrics, 199
problems with, 197
process improvement, 203-207
scaling, 200
cockpit of plane example, 44
code management. See source code
management
code promotion support in environment
configuration, 52
code regression, avoiding, 33
code variants. See variant management
collaboration, psychology of, 153
commercial tools, open source tools
versus, 21
Committee of Sponsoring Organizations
(COSO), 175-176
communicating release status, 80
communication styles, 147
active listening, 148-149
consultation methods, 148
gender differences in, 147-148
communications planning in deployment,
92-93
compile dependency management in build
gineering, 33-34
compliance, 168-172. See also best practices
Cobit framework, 176-177
configuration management assessments,
183-185
conformance versus, 192-193
COSO (Committee of Sponsoring
Organizations), 175-176
GAO (Government Accountability Of-
face) FDIC audit, 177-179
GAO (Government Accountability Of-
face) NARA audit, 179-181
goals of, 172-173
HIPAA (Health Insurance Portability
and Accountability Act of 1996),
177
importance of, 173
improving quality and productivity via, 183
moral argument for, 182-183
OCC (Office of the Comptroller of the
Currency), 181
requirements, 181-182
SOX (Sarbanes-Oxley Act of 2002),
174-175
starting point for, 173-174
compromisers, middle-borns as, 151
“conducting bakeoffs,” 42
configuration audits, xxxvi, 90, 191
configuration control, 62-63
defined, 190
release management and, 81
configuration identification, xxxvi, 190
configuration items (CIs), xxxiii, 6, 32
defined, 189-190
in hardware configuration manage-
ment, 107-108
release maps of, 77
configuration management. See CM (con-
figuration management)
configuration management assessments,
183-185
centralized environment CMDBs, 55-56
in ITIL framework, 211-213
configuration management driven develop-
ment (CMDD), 101
configuration management system (CMS)
in ITIL framework, 210
configuration management-driven develop-
development (CMDD), 101
configuration status accounting (CSA),
191
conflict between teams in change control,
65
conflict of interest in build engineering,
conformance, noncompliance versus, 192-193
consensus, failing to gain, 165
consultation methods for communication improvement, 148
continuous integration (CI), 40, 47-48, 216-217
coordination function, release management as, 80-81
copybranches in source code management, 12-13
corporate culture. See culture
COSO (Committee of Sponsoring Organizations), 175-176
cost of quality
for build engineering, 42
for source code management, 23-24
cryptography, signing release packages with, 82
CSA (configuration status accounting), 191
culture
matching process to, 127-128
personality and, 156-159
acceptance of others, 157
loose cannons, 157-158
standards, following, 156, 158
cVS, 4
database dependencies in environment configuration, 52-55
Davidson, James Duncan, 38
defect tracking in source code management, 16-17, 26
definitive media library (DML) in ITIL framework, 210
deltas in source code management, 12-13
Deming, W. Edwards, 71, 114, 119, 162, 163, 202
dependencies
in build engineering, 33-34
in hardware configuration management, 108
runtime dependencies. See environment configuration
deployment, 83-86. See also release management
best practices, 87-90
communications planning, 92-93
configuration audits, 90
of firmware changes, 109
goals of, 86
importance of, 87
improvements to, 93-94
interface control, 92
in ITIL framework, 210
principles of, 86-87
responsibility for, 93
smoke test, 92
staging process, 87-89
starting point for, 87
"trust, but verify," 93
deployment frameworks, 89
design documents in hardware configuration management, 107-108
dML (definitive media library) in ITIL framework, 210
DSM-IV R psychiatric diagnostic manual, 146

E
e-change control, 67
EIA. See ANSI/ITAA EIA-649-A standard
Electronic Records Archives (ERA) audit, 179-181
emergency change control, 64, 199
ter entry criteria in change control, 70
environment configuration, xxxiv, 48-50
best practices, 57
centralized environment CMDBs, 55-56
change control and, 56
code promotion support, 52
dependency management, 52-55
environment management in, 57
future of, 57-58
goals of, 50-51
importance of, 51
principles of, 51
starting point for, 51-52
environment management in environment configuration, 57
environment monitoring in ITIL framework, 212
ERA (Electronic Records Archives) audit, 179-181
ergonomics of release management, 77-80
Erikson, Erik, 144
errors. See human error
ethical issues in build engineering, 36-37
evaluating change control, 71. See also
selecting
exit criteria in change control, 70
extensibility of source code management
tools, 22

F
“failure is not an option,” 138-139
family dynamics in personality, 155
FDIC (Federal Deposit Insurance Corporation) audit, 177-179
feature branching in source code management, 12-13
Federal Deposit Insurance Corporation (FDIC) audit, 177-179
Feldman, Stuart, 38
firmware changes, deployment of, 109
firstborns as leaders, 150-151
forged approvals example (change control), 69
forgetting to ask for help, 166
fostering teamwork, 131
Fowler, Martin, 216
frameworks
Agile/SCRUM, 216-217
CMM/CMMI, 207-208
Cobit framework, 197-207
change control, 198-203
emergency change control, 199
metrics, 199
problems with, 197
process improvement, 203-207
scaling, 200
goals of, 188
importance of, 188
ITIL framework, 208-213
asset management, 210
change advisory board (CAB), 209
change control, 209
CMS (configuration management system), 210
configuration management database (CMDB), 211-213
definitive media library (DML), 210
evironment monitoring, 212
process improvement, 212
relationships among systems, 210-211
release management and deployment, 210
service asset and configuration management (SACM), 209
validation and testing, 213
Open Unified Process, 215-216
standards versus, 196-197
starting point for, 189
SWEBOK framework, 214-215
terminology, 189-193

G
gaining consensus, 165
GAO (Government Accountability Office)
FDIC audit, 177-179
NARA audit, 179-181
gatekeeping change control, 62
gender differences in communication styles, 147-148
globally distributed teams, source code management in, 17-18
GNU Make, 38
governance. See compliance
Government Accountability Office (GAO)
FDIC audit, 177-179
NARA audit, 179-181
graybox testing, 154
group dynamics, 154
guerrilla tactics for overcoming resistance to change, 138-139

H
hardware configuration management, 103-106
changes to firmware, deploying, 109
dependencies in, 108
future of, 109
goals of, 106
importance of, 106
interface control in, 108
starting point for, 107
traceability in, 108-109
version control in, 107-108
Health Insurance Portability and Accountability Act of 1996 (HIPAA), 177

hierarchy of change control, 67

HIPAA (Health Insurance Portability and Accountability Act of 1996), 177

honesty, need for, 168

human error, avoiding in release management, 78-79

I

IDEs (integrated development environments), 25, 40-41

IEEE 828 standard, 193-195

immutable version IDs
  in build engineering, 33
  in release management, 76-77

implementation time for source code management, 25

incremental changes in process improvement, 136

indecisiveness, 155

independent builds, 34-35

industrial psychology. See psychology

industry frameworks. See frameworks

industry standards. See standards

information processing preferences, 149-150

initiative of youngest-borns, 151

inner merges in source code management, 15

input from stakeholders, 132-133

integrated development environments (IDEs), 25, 40-41

interface control, 92
  defined, 190-191
  in hardware configuration management, 108

investment bank example (change control), 66-67

ISACA. See Cobit framework

ISO 10007 standard, 195

ISO/IEC/IEEE 12207 standard, 196

ISO/IEC/IEEE 15288 standard, 196

IT controls, 168-172. See also best practices

Cobit framework, 176-177

configuration management assessments, 183-185

COSO (Committee of Sponsoring Organizations), 175-176

GAO (Government Accountability Office) FDIC audit, 177-179

GAO (Government Accountability Office) NARA audit, 179-181

goals of, 172-173

HIPAA (Health Insurance Portability and Accountability Act of 1996), 177

importance of, 173

improving quality and productivity via, 183

moral argument for, 182-183

OCC (Office of the Comptroller of the Currency), 181

requirements, 181-182

SOX (Sarbanes-Oxley Act of 2002), 174-175

starting point for, 173-174

IT standards. See standards

ITIL framework, 208-213

asset management, 210

change advisory board (CAB), 209

change control, 209

CMS (configuration management system), 210

configuration management database (CMDB), 211-213

definitive media library (DML), 210

environment monitoring, 212

process improvement, 212

relationships among systems, 210-211

release management and deployment, 210

service asset and configuration management (SACM), 209

validation and testing, 213

itSMF. See ITIL framework

ivy tower, remaining in, 167

J-K

just-in-time process improvement, 120

L

language barriers, 78

leadership
  failing to show, 165
Index

of firstborns, 150-151
in process improvement, 133
Lean Software Development, 119-120
learning from mistakes. See mistakes,
learning from
lessons learned. See mistakes, learning from
lifecycle. See ISO/IEC/IEEE 12207 standard;
ISO/IEC/IEEE 15288 standard
listening to organizational rhythm, 134-136

M
MAC SHA1, 82
Make, 38
Maven, 38-39
MBI (Myers-Briggs Inventory), 144
MD5, 82
merging in source code management, 15-16
message verification, 148-149
metadata, 10
metrics in Cobit framework, 199
middle-borns as compromisers, 151
missing the big picture, 163-164
mistakes, learning from, 161-162
examples of mistakes, 163-168
becoming part of problem, 165-166
failing to gain consensus, 165
failing to show leadership, 165
lack of honesty, 168
missing big picture, 163-164
not asking for help, 166
promoting process improvement, 165
remaining in ivory tower, 167
writing release automation, 164
goals of, 162
importance of, 162
lessons learned, 166-167
starting point for, 162
understanding mistakes, 163
moral argument for IT controls, 182-183
Myers-Briggs Inventory (MBI), 144

N
NARA (National Archives and Records Administration) audit, 179-181
National Archives and Records Administration (NARA) audit, 179-181
nightly builds, CI (continuous integration) versus, 47-48
noncompliance, conformance versus, 192-193

O
OCC (Office of the Comptroller of the Currency), 181
OCEAN personality assessment, 144
Office of the Comptroller of the Currency (OCC), 181
only children, personality of, 151-152
open API for source code management tools, 22
open source tools, commercial tools versus, 21
Open Unified Process, 118-119, 215-216
operating systems, release management support, 82
Optimistic checkout model, 10
organizational structure in build engineering, 37-38
outer merges in source code management, 16
overengineering
in build engineering, 35-36
in process improvement, 120-121
of source code management tools, 22-23

P
packaging technology, understanding of, 78-79
personality. See also psychology; resistance to change
assessments for understanding, 144
birth order roles, 150-152
firstborns as leaders, 150-151
middle-borns as compromisers, 151
only children, 151-152
self-expression, 152
youngest as initiators, 151
communication styles, 147
active listening, 148-149
consultation methods, 148
Index

gender differences in, 147-148
defined, 143
family dynamics in, 155
goals of understanding, 142-143
information processing preferences, 149-150
workplace culture and, 156-159
acceptance of others, 157
loose cannons, 157-158
standards, following, 156, 158
Poppendieck, Mary and Tom, 119
procedural justice, 132
process consultation, 122
process engineering, 64
process improvement, 109-114
Agile, 118
blindness example, 116
CMMI (Capability Maturity Model Integrated), 116-117
in Cobit framework, 203-207
goals of, 114-115
importance of, 115
in ITIL framework, 212
just-in-time process improvement, 120
Lean Software Development, 119-120
Open Unified Process, 118-119
overengineering, avoiding, 120-121
process consultation, 122
promoting, 165
resistance to change, overcoming, 123-126
combining with technology training, 134-135
goals of, 126-127
guerrilla tactics for, 138
importance of, 127
improvement from within company, 129-130
incremental changes, 136
leadership, 133
legitimate opposition, 132
listening to organizational rhythm, 134-136
matching process to culture, 127-128
pick your battles, 131
practicality of processes, 133-134
procedural justice, 132
promoting process improvement, 137
psychology and, 129
self-interest, addressing, 137
stakeholder input, 132-133
starting point for, 127
teamwork, encouraging, 131
scene surveys, 130
as service, 137-138
SPIN (Software Process-Improvement Network), 115
starting point for, 115
sustainability of, 122
technology and, 121
test-driven process improvement (TDPI), 136
testing in, 121
too little process, 120
verbose processes, 115, 118
processing preferences, 149-150
processing speed, 149-150
processing styles, 149
product maturity for source code management tools, 21-22
productivity improvement via compliance, 183
promoting code in environment configuration, 52
promoting process improvement, 137, 165
psychology. See also personality
listening to organizational rhythm, 134-136
process improvement and, 129
workplace applications of, 152-155
collaboration, 153
group dynamics, 154
teamwork, 153
testers and build engineers in development team, 153-154
Q
quality improvement via compliance, 183
R
RCS, 4
regression. See code regression
release automation, writing, 164
release calendars, 80-81
release engineering, xxxiv
release management, 71-74. See also deployment
build engineering and, 79
configuration control and, 81
as coordination function, 80-81
ergonomics of, 77-80
future of, 81-82
goals of, 74
importance of, 75
in ITIL framework, 210
principles of, 74-75
requirements tracking, 81
starting point for, 75
version IDs, 76-77
release maps, 77
release status, communicating, 80
requirements for compliance, 181-182
requirements tracking
in release management, 81
in source code management, 16-17, 26
reserved checkouts, unreserved checkouts versus, 10-11
resistance to change, overcoming, 123-126. See also personality
combining with technology training, 134-135
goals of, 126-127
guerrilla tactics for, 138
importance of, 127
improvement from within company, 129-130
incremental changes, 136
leadership, 133
legitimate opposition, 132
listening to organizational rhythm, 134-136
matching process to culture, 127-128
pick your battles, 131
practicality of processes, 133-134
procedural justice, 132
promoting process improvement, 137
psychology and, 129
self-interest, addressing, 137
stakeholder input, 132-133
starting point for, 127
teamwork, encouraging, 131
rhythm of organization, listening to, 134-136
rightsizing CM processes. See process improvement
risks
balancing, 158
in change control, 69
in source code management, 25
RM. See release management
time dependencies. See environment configuration

S
Sachs, Benjamin K., 119
SACM (service asset and configuration management), 209
sandboxes, 11
Sarbanes-Oxley Act of 2002, 174-175
scaling, in Cobit framework, 200
SCAMPI (Standard CMMI Appraisal Method for Process Improvement), 185
SCCS, 4
scene surveys, 130
SCM (source code management). See source code management
scripts for source code management tools, 22-23
SCRUM, 216-217
selecting
build tools, 38-42
source code management tools, 19-23
self-expression in birth order roles, 152
self-managed teams, 42
senior management oversight of change control, 64-65
senior management support for source code management, 9
sensory modalities, 149
separation of controls in compliance requirements, 182
SEPG (software engineering process group), 122
service asset and configuration management (SACM), 209
services
build engineering as, 103
process improvement as, 137-138
source code management as, 103
signing release packages, 82
smoke test, 92
Software Configuration Management Patterns: Effective Teamwork, Practical Integration (Berczuk), 121
Software Engineering Body of Knowledge (SWEBOK), 214-215
software engineering process group (SEPG), 122
Software Process-Improvement Network (SPIN), 115
source code management, xxxiii-xxxiv
  in architecture development, 102-103
  baselines, 8-10
  changesets, 16
  defect and requirements tracking, 16-17, 26
  in globally distributed teams, 17-18
  goals of, 4-6
  implementation time, 25
  importance of, 6
  principles of, 6
  reserved versus unreserved checkouts, 10-11
  risks in, 25
  sandboxes, 11
  senior management support for, 9
  starting point for, 7-8
  support process for, 25-27
  tool selection, 19-23
  total cost of ownership, 23-24
  training, 24-25
  usage model definition, 25
  user empowerment, 27
  variant management, 11-12
  bugfixes, 12-13
  copybranches versus deltas, 12-13
  merging, 15-16
  streams, 14-15
  SOX (Sarbanes-Oxley Act of 2002), 174-175
  specialized change control, 67
  SPIN (Software Process-Improvement Network), 115
  staging process, 87-89
  stakeholder input, 132-133
  Standard CMMI Appraisal Method for Process Improvement (SCAMPI), 185
  standards
    ANSI/ITAA EIA-649-A standard, 196
    frameworks versus, 196-197
    goals of, 188
    IEEE 828 standard, 193-195
    importance of, 188
    ISO 10007 standard, 195
    ISO/IEC/IEEE 12207 standard, 196
    ISO/IEC/IEEE 15288 standard, 196
    personality and workplace culture, 156, 158
    starting point for, 189
    terminology, 184, 189-193
    static code analysis, 41
    status accounting, xxxvi
    streams in source code management, 14-15
    subcontractor control in CMM/CMMI, 208
    defined, 192
    support process for source code management, 25-27
    sustainability of process improvement, 122
    SWEBOK framework, 214-215
    system integration of change control, 70
    systems architecture. See architecture

T

Tannen, Deborah, 147-148
TDD (test-driven development), 101
TDPI (test-driven process improvement), 136
  team conflict in change control, 65
  teams, self-managed, 42
  teamwork
    encouraging, 131
    psychology of, 153
  technology, process improvement and, 121
  technology architecture in build engineering, importance of, 46-47
  technology training, process improvement and, 134-135
  test-driven builds, 43
  test-driven development (TDD), 101
  test-driven process improvement (TDPI), 136
  testing
    blackbox versus whitebox versus graybox, 154
    deployment, 92
    in ITIL framework, 213
    as part of development team, 153-154
    in process improvement, 121, 136
role in architecture and CM (configuration management), 99-101
third-party training, vendor training versus, 24
time management in change control, 66
token substitution, 54-55
tool selection
  in build engineering, 38-42
  in source code management, 19-23
total cost of quality
  for build engineering, 42
  for source code management, 23-24
traceability
  in compliance requirements, 182
  in hardware configuration management, 108-109
trading firm example (change control), 67
training
  in architecture development, 102-103
  in build engineering, 42
  importance of, 17, 23, 167
  in source code management, 24-25
technology training, process improvement and, 134-135
transparency in process improvement, 122
“trust, but verify,” 43-44, 93

U
unreserved checkouts, reserved checkouts versus, 10-11
usage model definition in source code management, 25
user empowerment in source code management, 27

V
validation, 136, 213
variant management, 11-12
  bugfixes, 12-13
  copybranches versus deltas, 12-13
  merging, 15-16
  streams, 14-15
vendor commitment for source code management tools, 21-22
vendor control, 192
vendor training, third-party training versus, 24

verbose processes, 115, 118
verification, 136
verifying the message, 148-149
version control in hardware configuration management, 107-108. See also source code management
version IDs
  in build engineering, 32-33
  in release management, 76-77

W-X
whitebox testing, 154
workplace applications of psychology, 152-155
collaboration, 153
  group dynamics, 154
teamwork, 153
testers and build engineers in development team, 153-154
workplace culture. See culture workspaces. See sandboxes

Y-Z
youngests as initiators, 151