Extracted from:

Manage It!

Your Guide to Modern, Pragmatic Project Management

Manage It!

Your Guide to Modern, Pragmatic Project Management

Johanna Rothman

The Pragmatic Bookshelf
Raleigh, North Carolina Dallas, Texas

Contents

Foreword **14**

Preface **16**

1 Starting a Project **19**
- 1.1 Define Projects and Project Managers 19
- 1.2 Manage Your Drivers, Constraints, and Floats 21
- 1.3 Discuss Your Project Constraints with Your Client or Sponsor 24
- 1.4 Decide on a Driver for Your Project 25
- 1.5 Manage Sponsors Who Want to Overconstrain Your Project 27
- 1.6 Write a Project Charter to Share These Decisions . . . 29
- 1.7 Know What Quality Means for Your Project 32

2 Planning the Project **35**
- 2.1 Start the Wheels Turning 35
- 2.2 Plan Just Enough to Start 36
- 2.3 Develop a Project Plan Template 37
- 2.4 Define Release Criteria 44
- 2.5 Use Release Criteria 49

3 Using Life Cycles to Design Your Project **52**
- 3.1 Understanding Project Life Cycles 52
- 3.2 Overview of Life Cycles 53
- 3.3 Seeing Feedback in the Project 57
- 3.4 Larger Projects Might Have Multiple Combinations of Life Cycles 58
- 3.5 Managing Architectural Risk 62
- 3.6 Paddling Your Way Out of a Waterfall 64
- 3.7 My Favorite Life Cycles 65

4 Scheduling the Project **66**
4.1 Pragmatic Approaches to Project Scheduling 66
4.2 Select from These Scheduling Techniques 68
4.3 Start Scheduling with a Low-Tech Tool 71

5 Estimating the Work **79**
5.1 Pragmatic Approaches to Project Estimation 79
5.2 Milestones Define Your Project's Chunks 93
5.3 How Little Can You Do? 95
5.4 Estimating with Multitasking 95
5.5 Scheduling People to Multitask by Design 96
5.6 Using Rolling-Wave Scheduling 97
5.7 Deciding on an Iteration Duration 98
5.8 Estimating Using Inch-Pebbles Wherever Possible . . . 100

6 Recognizing and Avoiding Schedule Games **103**
6.1 Bring Me a Rock 103
6.2 Hope Is Our Most Important Strategy 106
6.3 Queen of Denial 108
6.4 Sweep Under the Rug 111
6.5 Happy Date 113
6.6 Pants on Fire 115
6.7 Split Focus 117
6.8 Schedule Equals Commitment 119
6.9 We'll Know Where We Are When We Get There 121
6.10 The Schedule Tool Is Always Right 123
6.11 We Gotta Have It; We're Toast Without It 126
6.12 We Can't Say No 128
6.13 Schedule Chicken 130
6.14 90% Done 131
6.15 We'll Go Faster Now 133
6.16 Schedule Trance 135

7 Creating a Great Project Team **137**
7.1 Recruit the People You Need 137
7.2 Help the Team Jell 139
7.3 Make Your Organization Work for You 142
7.4 Know How Large a Team You Need 145
7.5 Know When to Add More People 147
7.6 Become a Great Project Manager 147
7.7 Know When It's Time to Leave 150

8 Steering the Project **158**
8.1 Steer the Project with Rhythm 158
8.2 Conduct Interim Retrospectives 159
8.3 Rank the Requirements 160
8.4 Timebox Requirements Work 163
8.5 Timebox Iterations to Four or Fewer Weeks 166
8.6 Use Rolling-Wave Planning and Scheduling 167
8.7 Create a Cross-Functional Project Team 170
8.8 Select a Life Cycle Based on Your Project's Risks . . . 171
8.9 Keep Reasonable Work Hours 172
8.10 Use Inch-Pebbles 173
8.11 Manage Interruptions 174
8.12 Manage Defects Starting at the Beginning of the Project 176

9 Maintaining Project Rhythm **181**
9.1 Adopt or Adapt Continuous Integration for Your Project 181
9.2 Create Automated Smoke Tests for the Build 183
9.3 Implement by Feature, Not by Architecture 184
9.4 Get Multiple Sets of Eyes on Work Products 189
9.5 Plan to Refactor 190
9.6 Utilize Use Cases, User Stories, Personas, and Scenarios to Define Requirements 192
9.7 Separate GUI Design from Requirements 193
9.8 Use Low-Fidelity Prototyping as Long as Possible . . . 194

10 Managing Meetings **196**
10.1 Cancel These Meetings 196
10.2 Conduct These Types of Meetings 199
10.3 Project Kickoff Meetings 200
10.4 Release Planning Meetings 200
10.5 Status Meetings 201
10.6 Reporting Status to Management 206
10.7 Project Team Meetings 207
10.8 Iteration Review Meetings 208
10.9 Troubleshooting Meetings 208
10.10 Manage Conference Calls with Remote Teams 210

11 Creating and Using a Project Dashboard **214**
11.1 Measurements Can Be Dangerous 214
11.2 Measure Progress Toward Project Completion 217
11.3 Develop a Project Dashboard for Sponsors 240
11.4 Use a Project Weather Report 243

12 Managing Multisite Projects **248**
12.1 What Does a Question Cost You? 249
12.2 Identify Your Project's Cultural Differences 250
12.3 Build Trust Among the Teams 251
12.4 Use Complementary Practices on a Team-by-Team Basis 254
12.5 Look for Potential Multisite Project and Multicultural Problems 262
12.6 Avoid These Mistakes When Outsourcing 264

13 Integrating Testing into the Project **267**
13.1 Start People with a Mind-Set Toward Reducing Technical Debt 267
13.2 Reduce Risks with Small Tests 268
13.3 TDD Is the Easiest Way to Integrate Testing into Your Project 269
13.4 Use a Wide Variety of Testing Techniques 272
13.5 Define Every Team Member's Testing Role 275
13.6 What's the Right Developer-to-Tester Ratio? 279
13.7 Make the Testing Concurrent with Development 285
13.8 Define a Test Strategy for Your Project 285
13.9 System Test Strategy Template 286
13.10 There's a Difference Between QA and Test 288

14 Managing Programs **290**
14.1 When Your Project Is a Program 290
14.2 Organizing Multiple Related Projects into One Release 291
14.3 Organizing Multiple Related Projects Over Time 293
14.4 Managing Project Managers 296
14.5 Creating a Program Dashboard 298

15 Completing a Project **300**
15.1 Managing Requests for Early Release 300
15.2 Managing Beta Releases 301
15.3 When You Know You Can't Meet the Release Date . . . 302
15.4 Shepherding the Project to Completion 310
15.5 Canceling a Project 314

16 Managing the Project Portfolio **317**
16.1 Build the Portfolio of All Projects 317
16.2 Evaluate the Projects 319
16.3 Decide Which Projects to Fund Now 320
16.4 Rank-Order the Portfolio 320
16.5 Start Projects Faster 321
16.6 Manage the Demand for New Features with a Product Backlog 323
16.7 Troubleshoot Portfolio Management 325

A More Detailed Information About Life Cycles **332**
A.1 Serial Life Cycle: Waterfall or Phase-Gate 332
A.2 Iterative Life Cycle: Spiral, Evolutionary Prototyping, Unified Process 336
A.3 Incremental Life Cycle: Staged Delivery, Design to Schedule 340
A.4 Agile Life Cycles 341

B Glossary of Terms **345**

C Bibliography **347**

Index **352**

Foreword

Hello, and welcome to Johanna's latest book. I'm currently a director at Yahoo! (in Berkeley) and have been in the software business for several decades. In fact, you might have heard of Digital Equipment Corporation (the foundation of the early Internet) and its Alpha system. That was a very important project for me.

I played a major role in the delivery of the Alpha software. It was a monumental task: some 2,000 engineers scattered all over the world, all working on various parts of the system. It required rigorous planning and project management, and we delivered on a four-year schedule within one month of our target date. So, as you can imagine, I thought I was a pretty good manager! But I was about to find out what an *excellent* manager is like.

In May 1996, I decided to leave DEC, and I heard about a job opening at another major software company in the Boston area. It was just the kind of challenge I relish, director of a product group—"a team living in chaos." Great, I thought. This is what I do! Coax the potential out of the chaos, and help deliver an actual working product. Now where's my white horse?

I heard that a consultant had been brought in to "triage" the development of the group's beta release. This only strengthened my conviction that they would soon find the hero they had been waiting for—in me.

But, wow! Instead, I was instantly (and progressively more and more) humbled and impressed. I understand what consultants are supposed to do. . . but when do they actually articulate situations in practical and actionable ways? This consultant had done just that. And in just a couple of months, she had managed to get all the pieces in place: a project charter, a program plan, and project plans, as well as defined roles and responsibilities, a defined development process, pertinent metrics, release criteria, beta customers. . . all the elements that are critical for a project to succeed.

But all that usually takes significant time to put in place—especially when starting from a deficit position. Yet here they were! You've probably guessed by now that this consultant was Johanna Rothman. (Johanna has a case study about our joint adventure on her website—only the names have been changed to protect the guilty!)

Over the years since I first met Johanna, I've run software development organizations in companies large and small. And on numerous occasions, I've engaged Johanna's services to help move my team to the next level. Her assessment process is rigorous and provides the solid footing that's required for effective project management. She tailors effective workshops on a multitude of topics—for me, she has done iterative project requirements, project management, and QA. I have hired her for interim management positions and for one-on-one coaching for people with varied skill sets. Johanna draws on a broad range of experience in a diversity of situations and organizations, and she always manages to provide solutions that are practical and realistic—solutions that can actually be implemented to solve key problems.

And so, this book is a real gift from Johanna.

She pulls knowledge from all her years on the front lines and presents the material in a cohesive way. The book provides you with the tools you need to analyze your own situation, build a framework and rational plan, and then execute. Johanna gives you lots of tips and examples of what works and what doesn't—and advice on how to avoid the rat holes. Even after years of project and program management experience, I learned new things when reviewing this book. And when I'm in a new or challenging situation or when I need a sounding board to help me think through a tough problem, Johanna is the one I call.

Oh, yeah—that project we worked on when I first met Johanna? We shipped the product to the beta customers, and it worked!

I know Johanna's book will help you succeed as well.

Ellen R. Salisbury (Director, Yahoo! Research Berkeley)
April 2007

Preface

You've been bombarded with a ton of techniques, practices, and unsolicited pieces of advice about how to manage projects. All of them are saying "Look at me, I'm right."

Well, many of them are right—under certain conditions. Since each project is unique, you will need to evaluate your context (the project, the project team, and the business in which you're working) and then make pragmatic choices about what will work and what won't.

Every day your projects become faster-paced, your customers grow more impatient, and there is less and less tolerance for products that don't work. What worked before might have been good enough to get you here, but the chances that it will work in the future are not good. You must take advantage of all practices and techniques to reduce your project's risk, including considering agile techniques for every project.

This book is a risk-based guide to making good decisions about how to plan and guide your projects. It will help software project managers, team members, and software managers succeed. Much of the information also applies if you are building more tangible products, such as a house or a circuit board, or if you are managing a service project.

I'm assuming you're managing a high-tech project, with at least some software component. You might have had some of the same project management experiences as I have: lots of software projects and some hardware/software combination projects. I've also managed a few service projects, such as planning and holding conferences. I've been part of some construction projects (one new house, one small remodel, and one large remodel). But the bulk of my experience is with software or software/hardware projects in some form.

It's harder to manage software projects than it is to manage projects that have a tangible deliverable. Software is ephemeral—not concrete, not material, not created out of substance—so we can't touch it, we

can't directly measure it, and we can't see it. It's harder to see the product unfold, and it's harder to see and anticipate the risks—so it's much harder to deal with risks. The way we practice software product development does not always help us see where the project is or where it's heading.

When you manage tangible-product projects, you can see the product take shape. You can see the shell of the building, the finish on the walls, and all the steps in between. With service products with a tangible result, such as a conference or meeting, you can gain some insight into the project if there are interim deliverables, such as rough-draft reports or run-throughs of meetings. Both tangible-product projects and some service projects allow you to see project progress before the end of the project.

So, what do you do when you can't directly see project progress? What do you do when you suspect the project smells funny, and you think it might be headed toward disaster? How do you deal with stakeholders who don't want to make the decisions that will help you create a successful project?

This book is about providing insight into your software projects and managing the risks that arise from within the project as well as the risks with which you start your projects. From chartering to release, each chapter discusses ways you can see inside your software project, measure it, feel it, taste it, and smell it.

One thing you won't find in this book is the One True Way to manage projects. There is no One True Way that works for all projects. You also won't find best practices. I'll suggest helpful practices for each life cycle that might help you and the project team achieve your goals.

You'll notice that there are forward and backward references in this book. That's because a project is a nonlinear system. Your early decisions for your current project have implications for how you'll finish this one—and possibly how you'll start the next one. How you manage projects might affect the way you can manage the product backlog or project portfolio.

All the templates in this book are also online, at the book's home page, http://pragmaticprogrammer.com/titles/jrpm.

I thank all the people who helped me write and edit this book: Tom Ayerst, Jim Bullock, Brian Burke, Piers Cawley, Shanti Chilukuri, Esther Derby, Michael F. Dwyer, Mark Druy, Jenn Greene, Payson Hall, Peter Harris, George Hawthorne, Ron Jeffries, Bil Kleb, Michael Lee, Hal Macomber, Rob McGurrin, Andrew McKinlay, Erik Petersen, Dwayne Phillips, Frederick Ros, Ellen Salisbury, George Stepanek, Andrew Wagner, and Jim Ward. My editor, Daniel Steinberg, provided exceptionally helpful feedback. Kim Wimpsett was again a copyeditor *par excellence*. I thank Steve Peter for his typesetting wizardry. Mark Tatro of Rotate Graphics developed all the schedule game cartoons. Working with Andy Hunt and Dave Thomas was, once again, my pleasure. Any mistakes are mine.

The stories I'm telling are all true—the names, companies, and specifics have all been changed to protect the innocent and the guilty.

Let's start.

Johanna Rothman
April 2007

A Pragmatic Career

Welcome to the Pragmatic Community. We hope you've enjoyed this title.

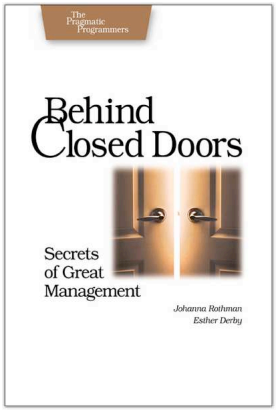
If you've enjoyed this book by Johanna Rothman, and want to advance your management career, you'll be interested in seeing what happens *Behind Closed Doors*. And see how you can lead you team to success by using *Agile Retrospectives*.

Behind Closed Doors

You can learn to be a better manager—even a great manager—with this guide. You'll find powerful tips covering:

• Delegating effectively • Using feedback and goal-setting • Developing influence • Handling one-on-one meetings • Coaching and mentoring • Deciding what work to do-and what not to do • . . . and more!

Behind Closed Doors Secrets of Great Management
Johanna Rothman and Esther Derby
(192 pages) ISBN: 0-9766940-2-6. $24.95
http://pragmaticprogrammer.com/titles/rdbcd

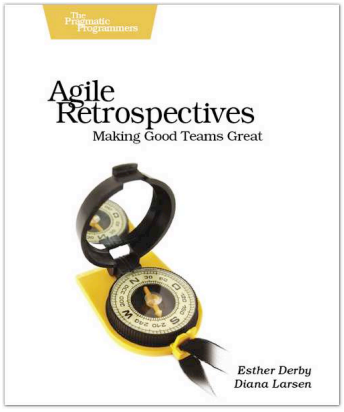


Agile Retrospectives

Mine the experience of your software development team continually throughout the life of the project. Rather than waiting until the end of the project—as with a traditional retrospective, when it's too late to help—agile retrospectives help you adjust to change *today*.

The tools and recipes in this book will help you uncover and solve hidden (and not-so-hidden) problems with your technology, your methodology, and those difficult "people issues" on your team.

Agile Retrospectives: Making Good Teams Great
Esther Derby and Diana Larsen
(170 pages) ISBN: 0-9776166-4-9. $29.95
http://pragmaticprogrammer.com/titles/dlret

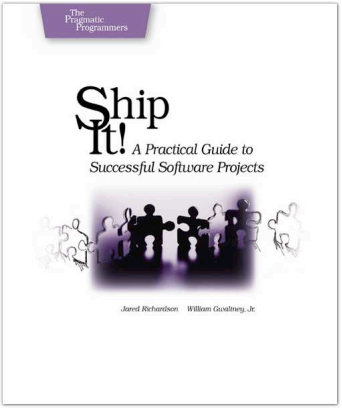


Competitive Edge

Need to get software out the door? Then you want to see how to *Ship It!* with less fuss and more features. And every developer can benefit from the *Practices of an Agile Developer*.

Ship It!

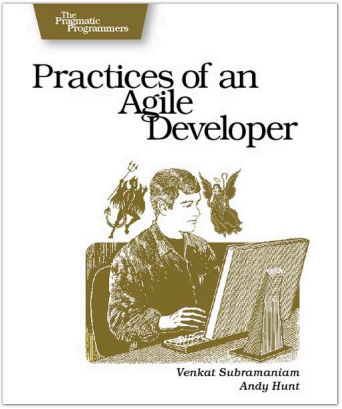
Page after page of solid advice, all tried and tested in the real world. This book offers a collection of tips that show you what tools a successful team has to use, and how to use them well. You'll get quick, easy-to-follow advice on modern techniques and when they should be applied. **You need this book if:** • You're frustrated at lack of progress on your project. • You want to make yourself and your team more valuable. • You've looked at methodologies such as Extreme Programming (XP) and felt they were too, well, extreme. • You've looked at the Rational Unified Process (RUP) or CMM/I methods and cringed at the learning curve and costs. • **You need to get software out the door without excuses**



Ship It! A Practical Guide to Successful Software Projects
Jared Richardson and Will Gwaltney
(200 pages) ISBN: 0-9745140-4-7. $29.95
http://pragmaticprogrammer.com/titles/prj

Practices of an Agile Developer

Agility is all about using feedback to respond to change. Learn how to apply the principles of agility throughout the software development process • Establish and maintain an agile working environment • Deliver what users really want • Use personal agile techniques for better coding and debugging • Use effective collaborative techniques for better teamwork • Move to an agile approach



Practices of an Agile Developer: Working in the Real World
Venkat Subramaniam and Andy Hunt
(189 pages) ISBN: 0-9745140-8-X. $29.95
http://pragmaticprogrammer.com/titles/pad

Cutting Edge

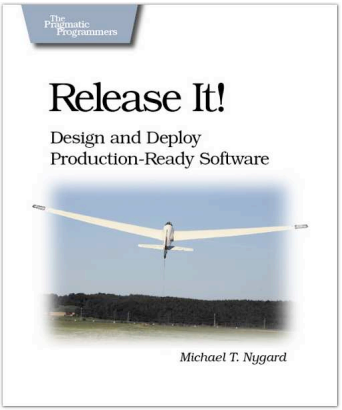
Now that you've finished your project, are you sure that it's ready for the real world? Are you truly ready to *Release It!* in this crazy world?

Interested in Ruby on Rails, but don't want to learn another framework from scratch? You don't have to! *Rails for Java Programmers*leverages you and your team's knowledge of Java to quickly learn the Rails environment.

Release It!

Whether it's in Java, .NET, or Ruby on Rails, getting your application ready to ship is only half the battle. Did you design your system to survive a sudden rush of visitors from Digg or Slashdot? Or an influx of real world customers from 100 different countries? Are you ready for a world filled with flakey networks, tangled databases, and impatient users?

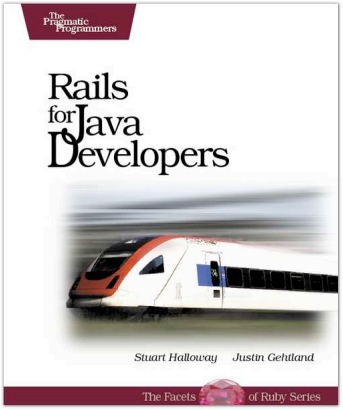
If you're a developer and don't want to be on call at 3AM for the rest of your life, this book will help.



Design and Deploy Production-Ready Software
Michael T. Nygard
(368 pages) ISBN: 0-9787392-1-3. $34.95
http://pragmaticprogrammer.com/titles/mnee

Rails for Java Developers

Enterprise Java developers already have most of the skills needed to create Rails applications. They just need a guide which shows how their Java knowledge maps to the Rails world. That's what this book does. It covers: • The Ruby language • Building MVC Applications • Unit and Functional Testing • Security • Project Automation • Configuration • Web Services This book is the fast track for Java programmers who are learning or evaluating Ruby on Rails.



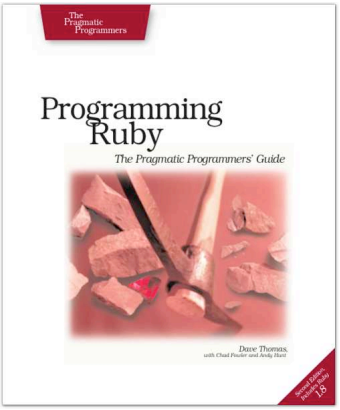
Rails for Java Developers
Stuart Halloway and Justin Gehtland
(300 pages) ISBN: 0-9776166-9-X. $34.95
http://pragmaticprogrammer.com/titles/fr_r4j

Facets of Ruby Series

If you're serious about Ruby, you need the definitive reference to the language. The Pickaxe: *Programming Ruby: The Pragmatic Programmer's Guide, Second Edition*. This is *the* definitive guide for all Ruby programmers. And you'll need a good text editor, too. On the Mac, we recommend TextMate.

Programming Ruby (The Pickaxe)

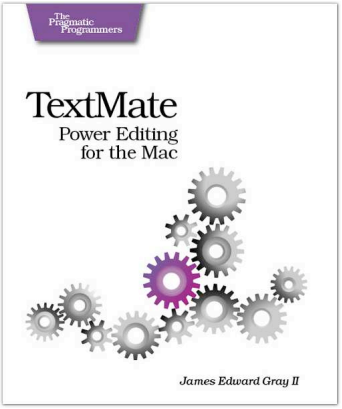
The Pickaxe book, named for the tool on the cover, is the definitive reference to this highly-regarded language. • Up-to-date and expanded for Ruby version 1.8 • Complete documentation of all the built-in classes, modules, and methods • Complete descriptions of all ninety-eight standard libraries • 200+ pages of new content in this edition • Learn more about Ruby's web tools, unit testing, and programming philosophy



Programming Ruby: The Pragmatic Programmer's Guide, 2nd Edition
Dave Thomas with Chad Fowler and Andy Hunt
(864 pages) ISBN: 0-9745140-5-5. $44.95
http://pragmaticprogrammer.com/titles/ruby

TextMate

If you're coding Ruby or Rails on a Mac, then you owe it to yourself to get the TextMate editor. And, once you're using TextMate, you owe it to yourself to pick up this book. It's packed with information which will help you automate all your editing tasks, saving you time to concentrate on the important stuff. Use snippets to insert boilerplate code and refactorings to move stuff around. Learn how to write your own extensions to customize it to the way you work.



TextMate: Power Editing for the Mac
James Edward Gray II
(200 pages) ISBN: 0-9787392-3-X. $29.95
http://pragmaticprogrammer.com/titles/textmate

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style, and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

Manage It! Home Page
http://pragmaticprogrammer.com/titles/jrpm
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragmaticprogrammer.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragmaticprogrammer.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragmaticprogrammer.com/news
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragmaticprogrammer.com/titles/jrpm.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	www.pragmaticprogrammer.com/catalog
Customer Service:	orders@pragmaticprogrammer.com
Non-English Versions:	translations@pragmaticprogrammer.com
Pragmatic Teaching:	academic@pragmaticprogrammer.com
Author Proposals:	proposals@pragmaticprogrammer.com