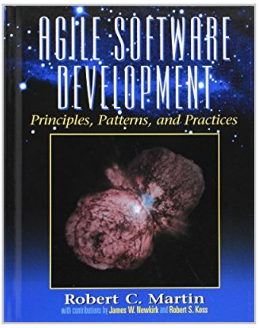
AGILE SOFTWARE DEVELOPMENT
Principles, Patterns, and Practices
Robert C. Martin
with contributions by James W. Newkirk and Robert S. Koss

Agile Software Development, Principles, Patterns, and Practices

By Robert C. Martin

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin

Written *by* a software developer for software developers, this book is a unique collection of the latest software development methods. The author includes OOD, UML, Design Patterns, Agile and XP methods with a detailed description of a complete software design for reusable programs in C++ and Java. Using a practical, problem-solving approach, it shows how to develop an object-oriented application—from the early stages of analysis, through the low-level design and into the implementation. Walks readers through the designer's thoughts — showing the errors, blind alleys, and creative insights that occur throughout the software design process. The book covers: Statics and Dynamics; Principles of Class Design; Complexity Management; Principles of Package Design; Analysis and Design; Patterns and Paradigm Crossings. Explains the principles of OOD, one by one, and then demonstrates them with numerous examples, completely worked-through designs, and case studies. Covers traps, pitfalls, and work arounds in the application of C++ and OOD and then shows how Agile methods can be used. Discusses the methods for designing and developing big software in detail. Features a three-chapter, in-depth, single case study of a building security system. For Software Engineers, Programmers, and Analysts who want to understand how to design object oriented software with state of the art methods.

Agile Software Development, Principles, Patterns, and Practices

By Robert C. Martin

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin

Written *by* a software developer for software developers, this book is a unique collection of the latest software development methods. The author includes OOD, UML, Design Patterns, Agile and XP methods with a detailed description of a complete software design for reusable programs in C++ and Java. Using a practical, problem-solving approach, it shows how to develop an object-oriented application—from the early stages of analysis, through the low-level design and into the implementation. Walks readers through the designer's thoughts — showing the errors, blind alleys, and creative insights that occur throughout the software design process. The book covers: Statics and Dynamics; Principles of Class Design; Complexity Management; Principles of Package Design; Analysis and Design; Patterns and Paradigm Crossings. Explains the principles of OOD, one by one, and then demonstrates them with numerous examples, completely worked-through designs, and case studies. Covers traps, pitfalls, and work arounds in the application of C++ and OOD and then shows how Agile methods can be used. Discusses the methods for designing and developing big software in detail. Features a three-chapter, in-depth, single case study of a building security system. For Software Engineers, Programmers, and Analysts who want to understand how to design object oriented software with state of the art methods.

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin Bibliography

- Sales Rank: #102463 in Books
- Brand: Martin, Robert Cecil
- Published on: 2002-10-25
- Original language: English
- Number of items: 1
- Dimensions: 10.10" h x 1.00" w x 8.00" l, 2.20 pounds
- Binding: Hardcover
- 529 pages

Editorial Review

From the Back Cover

Best selling author and world-renowned software development expert Robert C. Martin shows how to solve the most challenging problems facing software developers, project managers, and software project leaders today.

- Teaches software developers and project managers how to get projects done on time, and on budget using the power of Agile Development.
- Uses real-world case studies to show how to of plan, test, refactor, and pair program using eXtreme programming.
- Contains a wealth of reusable C++ and Java code.
- Focuses on solving customer oriented systems problems using UML and Design Patterns.

Robert C. Martin is President of Object Mentor Inc. Martin and his team of software consultants use Object-Oriented Design, Patterns, UML, Agile Methodologies, and eXtreme Programming with worldwide clients. He is the author of the best-selling book *Designing Object-Oriented C++ Applications Using the Booch Method* (Prentice Hall, 1995), Chief Editor of, *Pattern Languages of Program Design 3* (Addison Wesley, 1997), Editor of, *More C++ Gems* (Cambridge, 1999), and co-author of *XP in Practice*, with James Newkirk (Addison-Wesley, 2001). He was Editor in Chief of the *C++ Report* from 1996 to 1999. He is a featured speaker at international conferences and trade shows.

About the Author

ROBERT C. MARTIN is President of Object Mentor Inc. Martin and his team of software consultants use Object-Oriented Design, Patterns, UML, Agile Methodologies, and eXtreme Programming with worldwide clients. He is the author of the best-selling book *Designing Object-Oriented C++ Applications Using the Booch Method* (Prentice Hall, 1995), Chief Editor of, *Pattern Languages of Program Design 3* (Addison Wesley, 1997), Editor of, *More C++ Gems* (Cambridge, 1999), and co-author of *XP in Practice*, with James Newkirk (Addison-Wesley, 2001). He was Editor in Chief of the *C++ Report* from 1996 to 1999. He is a featured speaker at international conferences and trade shows.

Agile development is the ability to develop software quickly, in the face of rapidly changing requirements. In order to achieve this agility, we need to employ practices that provide the necessary discipline and feedback. We need to employ design principles that keep our software flexible and maintainable, and we need to know the design patterns that have been shown to balance those principles for specific problems. This book is an attempt to knit all three of these concepts together into a functioning whole.

This book describes those principles, patterns, and practices and then demonstrates, how they are applied by walking through dozens of different case studies. More importantly, the case studies are not presented as complete works. Rather, they are designs *in progress.* You will see the designers make mistakes, and you will observe how they identify the mistakes and eventually correct them. You will see them puzzle over conundrums and worry over ambiguities and trade-offs. You will see the *act* of design.

The Devil Is in the Details

This book contains a *lot* of Java and C++ code. I hope you will carefully read that code since, to a large degree, the code is the point of the book. The code is the actualization of what this book 6~ " has to say.

There is a repeating pattern to this book. It consists of a series of case studies of varying sizes. Some are very small, and some require several chapters to describe. Each case study is preceded by /material that is meant to prepare you for it. For example, the Payroll case study is preceded by chapters describing the object-oriented design principles and patterns used in the case study.

The book begins with a discussion of development practices and processes. That discussion is punctuated by a number of small case studies and examples. From there, the book moves on to the topic of design and design principles, and then to some design patterns, more design principles that govern packages, and more patterns. All of these topics are accompanied by case studies.

So prepare yourself to read some code and to pore over some UML diagrams. The book you are about to read is *very* technical, and its lessons, like the devil, are in the details.

A Little History

Over six years ago, I wrote a book entitled *Designing Object-Oriented C++ Applications using the Booch Method.* It was something of magnum opus for me, and I was very pleased with the result and with the sales.

This book started out as a second edition to *Designing,* but that's not how it turned out. Very little remains of the original book in these pages. Little more than three chapters have been carried through, and those chapters have been massively changed. The intent, spirit, and many of the lessons of the book are the same. And yet, I've learned a tremendous amount about software design and development in the six years since *Designing* came out. This book reflects that learning.

What a half-decade! *Designing* came out just before the Internet collided with the planet. Since then, the number of abbreviations we have to deal with has doubled. We have Design Patterns, Java, EJB, RMI, J2EE, XML, XSLT, HTML, ASP, JSP, Servlets, Application Servers, ZOPE, SOAP, C#, .NET, etc., etc. Let me tell you, it's been hard to keep the chapters of this book reasonably current!

The Booch Connection

In 1997, I was approached by Grady Booch to help write the third edition of his amazingly successful *Object-Oriented Analysis and Design with Applications.* I had worked with Grady before on some projects, and I had been an avid reader and contributor to his various works, including UML. So I accepted with glee. I asked my good friend Jim Newkirk to help out with the project.

Over the next two years, Jim and I wrote a number of chapters for the Booch book. Of course, that effort meant that I could not put as much effort into this book as I would have liked, but I felt that the Booch book was worth the contribution. Besides, this book was really just a second edition of *Designing* at the time, and my heart wasn't in it. If I was going to say something, I wanted to say something new and different.

Unfortunately, that version of the Booch book was not to be. It is hard to find the time to write a book during normal times. During the heady days of the ".com" bubble, it was nearly impossible. Grady got ever busier with Rational and with new ventures like Catapulse. So the project stalled. Eventually, I asked Grady and Addison Wesley if I could have the chapters that Jim and I wrote to include in this book. They graciously agreed. So several of the case study and UML chapters came from that source.

The Impact of Extreme Programming

In late 1998, XP reared its head and challenged our cherished beliefs about software development. Should we create lots of UML diagrams prior to writing any code, or should we eschew any kind of diagrams and just write lots of code? Should we write lots of narrative documents that describe our design, or should we try to make the code narrative and expressive so that ancillary documents aren't necessary? Should we program in pairs? Should we write tests before we write production code? What should we do?

This revolution came at an opportune time for me. During the middle to late 90s, Object Mentor was helping quite a few companies with object-oriented (OO) design and project management issues. We were helping companies get their projects *done.* As part of that help, we instilled our own attitudes and practices into the teams. Unfortunately, these attitudes and practices were not written down. Rather, they were an oral tradition that was passed from us to our customers.

By 1998, I realized that we needed to write down our process and practices so that we could better articulate them to our customers. So, I wrote many articles about process in the *C++ Report.* These articles missed the mark. They were informative, and in some cases entertaining, but instead of codifying the practices and attitudes that we actually used in our projects, they were an unwitting compromise to values that had been imposed upon me for decades. It took Kent Beck to show me that.

The Beck Connection

In late 1998, as I was fretting over codifying the Object-Mentor process, I ran into Kent's work on Extreme Programming (XP). The work was scattered through Ward Cunningham's *wiki* and was mixed with the writings oil many others. Still, with some work and diligence I was able to get the gist of what Kent was talking about. I was intrigued, but skeptical. Some of the things that XP talked about were exactly on target for my concept of a development process. Other things, however, like the lack of an articulated design step, left me puzzled.

Kent and I could not have come from more disparate software circumstances. He was a recognized Smalltalk consultant, and I was a recognized C++ consultant. Those two worlds found it difficult to communicate with one' another. There was an almost Kuhnian paradigm gulf between them.

Under other circumstances, I would never have asked Kent to write an article for the *C++ Report.* But the congruence of our thinking about process was able to breech the language gulf. In February of 1999, I met Kent in Munich at the OOP conference. He was giving a talk on XP in the room across from where I was giving a talk on principles of OOD. Being unable to hear that talk, I sought Kent out at lunch. We talked about XP, and I asked him to write an article for the *C++ Report.* It was a great article about an incident in which Kent and a coworker had been able to make a sweeping design change in a live system in a matter of an hour or so.

Over the next several months, I went through the slow process of sorting out my own fears about XP My greatest fear was in adopting a process in which there is no explicit up-front design step. I found myself balking at that. Didn't I have an obligation to my clients, and to the industry as a whole, to teach them that design is important enough to spend time on?

Eventually, I realized that I did not really practice such a step myself. Even in all the articles and books I had written about design, Booch diagrams, and UML diagrams, I had always used code as a way to verify that the diagrams were meaningful. In all my customer consulting, I would spend an hour or two helping them to draw diagrams and then I would direct them to explore those diagrams with code. I came to understand that though XP's words about design were foreign (in a Kuhnian sense), the practices behind the words were familiar to me.

My other fears about XP were easier to deal with. I had always been a closet pair programmer. XP gave me a way to come out of the closet and revel in my desire to program with a partner. Refactoring, continuous integration, and customer on-site were all very easy for me to accept. They were very close to the way I already advised my customers to work.

One practice of XP was a revelation for me. Test-first design sounds innocuous when you first hear it. It says to write test cases before you write production code. All production code is written to make failing test cases pass. I was not prepared for the profound ramifications that writing code this way would have. This practice has completely transformed the way I write software, and transformed it for the better. You can see that transformation in this book. Some of the code written in this book was written before 1999. You won't find test cases for that code. On the other hand, all of the code written after 1999 is presented with test cases, and the test cases are typically presented first. I'm sure you'll note the difference.

So, by the fall of 1999 I was convinced that Object Mentor should adopt XP as its process of choice and that I should let go of my desire to write my own process. Kent had done an excellent job of articulating the practices and process of XP, and my own feeble attempts paled in comparison.

Organization

This book is organized into six major sections followed by several appendices.

- Section 1: *Agile Development*
  This section describes the concept of agile development. It starts with the Manifesto of the Agile Alliance, provides an overview of Extreme Programming (XP), and then goes into many small case studies that illuminate some of the individual XP practices—especially those that have an impact upon the way we design and write code.
- Section 2: *Agile Design*
  The chapters in this section talk about object-oriented software design. The first chapter asks the question, *What is Design?* It discusses the problem of, and techniques for, managing complexity. Finally, the section culminates with the *principles of object-oriented class design.*
- Section 3: *The Payroll Case Study*
  This is the largest and most complete case study in the book. It describes the object-oriented design and C++ implementation of a simple batch payroll system. The first few chapters in this section describe the design patterns that the case study encounters. The final two chapters contain the full case study.
- Section 4: *Packaging the Payroll System*
  This section begins by describing the *principles of object-oriented package design.* It then goes on to illustrate those principles by incrementally packaging the classes from the previous section.
- Section 5: *The Weather Station Case Study*
  This section contains one of the case studies that was originally planned for the Booch book. The Weather Station study describes a company that has made a significant business decision and explains how the Java development team responds to it. As usual, the section begins with a description of the design patterns that will be used and then culminates in the description of the design and implementation.
- Section 6: *The ETS Case Study*

This section contains a description of an actual project that the author participated in. This project has been in production since 1999. It is the automated test system used to deliver and score the registry examination for the National Council of Architectural Registration Boards.

- UML Notation Appendices:
  The first two appendices contains several small case studies that are used to describe the UML notation.
- Miscellaneous Appendices

How to Use This Book

If You are a Developer...

Read the book cover to cover. This book was written primarily for developers, and it contains the information you need to develop software in an agile manner. Reading the book cover to cover introduces practices, then principles, then patterns, and then it provides case studies that tie them all together. Integrating all this knowledge will help you get your projects *done.*

If You Are a Manager or Business Analyst...

Read Section 1, *Agile Development.* The chapters in this section provide an in-depth discussion of agile principles and practices. They'll take you from requirements to planning to testing, refactoring, and programming. It will give you guidance on how to build teams and manage projects. It will help you get your projects *done.*

If You Want to Learn UML...

First read Appendix A, *UML Notation 1: The CGI Example.* Then read Appendix B, *UML Notation II: The STATMUX.* Then, read all the chapters in Section 3, *The Payroll Case Study.* This course of reading will give you a good grounding in both the syntax and use of UML. It will also help you translate between UML and a programming language like Java or C++.

If You Want to Learn Design Patterns...

To find a particular pattern, use the "List of Design Patterns" on page xxii to find the pattern you are interested in.

To learn about patterns in general, read Section 2, *Agile Design* to first learn about design principles, and then read Section 3, *The Payroll Case Study;* Section 4, *Packaging the Payroll System;* Section 5, *The Weather Station Case Study;* and Section 6, *The ETS Case Study.* These sections define all the patterns and show how to use them in typical situations.

If You Want to Learn about Object-Oriented Design Principles...

Read Section 2, *Agile Design;* Section 3, *The Payroll Case Study;* and Section 4, *Packaging the Payroll System.* These chapters will describe the principles of object-oriented design and will show you how to use them.

If You Want to Learn about Agile Development Methods...

Read Section 1, *Agile Development.* This section describes agile development from requirements to planning, testing, refactoring, and programming.

If You Want a Chuckle or Two...

Read Appendix C, *A Satire of Two Companies.*

Users Review

From reader reviews:

Peter Burnett:

Inside other case, little folks like to read book Agile Software Development, Principles, Patterns, and Practices. You can choose the best book if you like reading a book. So long as we know about how is important the book Agile Software Development, Principles, Patterns, and Practices. You can add knowledge and of course you can around the world by a book. Absolutely right, simply because from book you can recognize everything! From your country until foreign or abroad you can be known. About simple thing until wonderful thing you can know that. In this era, we are able to open a book or searching by internet system. It is called e-book. You may use it when you feel bored to go to the library. Let's read.

Mike Huey:

Do you certainly one of people who can't read pleasurable if the sentence chained within the straightway, hold on guys this kind of aren't like that. This Agile Software Development, Principles, Patterns, and Practices book is readable through you who hate those perfect word style. You will find the facts here are arrange for enjoyable examining experience without leaving actually decrease the knowledge that want to provide to you. The writer of Agile Software Development, Principles, Patterns, and Practices content conveys prospect easily to understand by lots of people. The printed and e-book are not different in the content material but it just different by means of it. So , do you continue to thinking Agile Software Development, Principles, Patterns, and Practices is not loveable to be your top list reading book?

Michael Quintanar:

A lot of people always spent their very own free time to vacation or maybe go to the outside with them friends and family or their friend. Do you realize? Many a lot of people spent they will free time just watching TV, or maybe playing video games all day long. If you wish to try to find a new activity honestly, that is look different you can read some sort of book. It is really fun for you personally. If you enjoy the book which you read you can spent the whole day to reading a reserve. The book Agile Software Development, Principles, Patterns, and Practices it is rather good to read. There are a lot of people that recommended this book. They were enjoying reading this book. In the event you did not have enough space to create this book you can buy typically the e-book. You can m0ore effortlessly to read this book out of your smart phone. The price is not too expensive but this book has high quality.

Shelley Gavin:

Reading can called thoughts hangout, why? Because if you find yourself reading a book mainly book entitled Agile Software Development, Principles, Patterns, and Practices your mind will drift away trough every dimension, wandering in every single aspect that maybe not known for but surely can be your mind friends. Imaging every single word written in a reserve then become one form conclusion and explanation that will maybe you never get prior to. The Agile Software Development, Principles, Patterns, and Practices giving you yet another experience more than blown away your mind but also giving you useful info for your better life on this era. So now let us show you the relaxing pattern at this point is your body and mind will be pleased when you are finished reading through it, like winning a casino game. Do you want to try this extraordinary shelling out spare time activity?

Download and Read Online Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin #BREWS03L45F

Read Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin for online ebook

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin books to read online.

Online Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin ebook PDF download

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin Doc

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin Mobipocket

Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin EPub

BREWS03L45F: Agile Software Development, Principles, Patterns, and Practices By Robert C. Martin