

## The Robert C Martin Clean Code Collection Collection Robert C Martin Series

*Shows how to bring unprecedented levels of professionalism and discipline to agile development - and thereby write far more effective, successful software*

*Presents practical advice on the disciplines, techniques, tools, and practices of computer programming and how to approach software development with a sense of pride, honor, and self-respect.*

*The Robert C. Martin Clean Code Collection consists of two bestselling eBooks: Clean Code: A Handbook of Agile Software Craftmanship The Clean Coder: A Code of Conduct for Professional Programmers In Clean Code, legendary software expert Robert C. Martin has teamed up with his colleagues from Object Mentor to distill their best agile practice of cleaning code "on the fly" into a book that will instill within you the values of a software craftsman and make you a better programmer--but only if you work at it. You will be challenged to think about what's right about that code and what's wrong with it. More important, you will be challenged to reassess your professional values and your commitment to your craft. In The Clean Coder, Martin introduces the disciplines, techniques, tools, and practices of true software craftsmanship. This book is packed with practical advice--about everything from estimating and coding to refactoring and testing. It covers much more than technique: It is about attitude. Martin shows how to approach software development with honor, self-respect, and pride; work well and work clean; communicate and estimate faithfully; face difficult decisions with clarity and honesty; and understand that deep knowledge comes with a responsibility to act. Readers of this collection will come away understanding How to tell the difference between good and bad code How to write good code and how to transform bad code into good code How to create good names, good functions, good objects, and good classes How to format code for maximum readability How to implement complete error handling without obscuring code logic How to unit test and practice test-driven development What it means to behave as a true software craftsman How to deal with conflict, tight schedules, and unreasonable managers How to get into the flow of coding and get past writer's block How to handle unrelenting pressure and avoid burnout How to combine enduring attitudes with new development paradigms How to manage your time and avoid blind alleys, marshes, bogs, and swamps How to foster environments where programmers and teams can thrive When to say "No"--and how to say it When to say "Yes"--and what yes really means*

*Presents guidelines on the art of coding with Perl, covering such topics as naming conventions, data and control structures, program decomposition, interface design, and error handling.*

*Practical Speech for Modern Business*

*Clean Architecture*

*The Passionate Programmer*

*Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software*

*A Craftsman's Guide to Software Structure and Design*

*A Handbook of Agile Software Craftmanship*

*The Software Craftsman*

***A single dramatic software failure can cost a company millions of dollars - but can be avoided with simple changes to design and architecture. This new edition of the best-selling industry standard shows you how to create systems that run longer, with fewer failures, and recover better when bad things happen. New coverage includes DevOps, microservices, and cloud-native architecture. Stability antipatterns have grown to include systemic problems in large-scale systems. This is a must-have pragmatic guide to engineering for production systems. If you're a software developer, and you don't want to get alerts every night for the rest of your life, help is here. With a combination of case studies about huge losses - lost revenue, lost reputation, lost time, lost opportunity - and practical, down-to-earth advice that was all gained through painful experience, this book helps you avoid the pitfalls that cost companies millions of dollars in downtime and reputation. Eighty percent of project life-cycle cost is in production, yet few books address this topic. This updated edition deals with the production of today's systems - larger, more complex, and heavily virtualized - and includes information on chaos engineering, the discipline of applying randomness and deliberate stress to reveal systematic problems. Build systems that survive the real world, avoid downtime, implement zero-downtime upgrades and continuous delivery, and make cloud-native applications resilient. Examine ways to architect, design, and build software - particularly distributed systems - that stands up to the typhoon winds of a flash mob, a Slashdotting, or a link on Reddit. Take a hard look at software that failed the test and find ways to make sure your software survives. To skip the pain and get the experience...get this book.***

***Widely considered one of the best practical guides to programming, Steve McConnell's original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help you: Design for minimum complexity and maximum creativity Reap the benefits of collaborative development Apply defensive programming techniques to reduce and flush out errors Exploit opportunities to refactor—or evolve—code, and do it safely Use construction practices that are right-weight for your project Debug problems quickly and effectively Resolve critical construction issues early and correctly Build quality into the beginning, middle, and end of your project Success in today's IT environment requires you to view your career as a business endeavor. In this book, you'll learn how to become an entrepreneur, driving your career in the direction of your choosing. You'll learn how to build your software development career step by step, following the same path that you would follow if you were building, marketing, and selling a product. After all, your skills themselves are a product. The choices you make about which technologies to focus on and which business domains to master have at least as much impact on your success as your technical knowledge itself--don't let those choices be accidental. We'll walk through all aspects of the decision-making process, so you can ensure that you're investing your time and energy in the right areas. You'll develop a structured plan for keeping your mind engaged and your skills fresh. You'll learn how to assess your skills in terms of where they fit on the value chain, driving you away from commodity skills and toward those that are in high demand. Through a mix of high-level, thought-provoking essays and tactical "Act on It" sections, you will come away with concrete plans you can put into action immediately. You'll also get a chance to read the perspectives of several highly successful members of our***

**industry from a variety of career paths. As with any product or service, if nobody knows what you're selling, nobody will buy. We'll walk through the often-neglected world of marketing, and you'll create a plan to market yourself both inside your company and to the industry in general. Above all, you'll see how you can set the direction of your career, leading to a more fulfilling and remarkable professional life.**

**Methods for managing complex software construction following the practices, principles and patterns of Domain-Driven Design with code examples in C# This book presents the philosophy of Domain-Driven Design (DDD) in a down-to-earth and practical manner for experienced developers building applications for complex domains. A focus is placed on the principles and practices of decomposing a complex problem space as well as the implementation patterns and best practices for shaping a maintainable solution space. You will learn how to build effective domain models through the use of tactical patterns and how to retain their integrity by applying the strategic patterns of DDD. Full end-to-end coding examples demonstrate techniques for integrating a decomposed and distributed solution space while coding best practices and patterns advise you on how to architect applications for maintenance and scale. Offers a thorough introduction to the philosophy of DDD for professional developers Includes masses of code and examples of concept in action that other books have only covered theoretically Covers the patterns of CQRS, Messaging, REST, Event Sourcing and Event-Driven Architectures Also ideal for Java developers who want to better understand the implementation of DDD**

**Domain Modeling Made Functional**

**More Effective Agile**

**Perl Best Practices**

**Tackle Software Complexity with Domain-Driven Design and F#**

**A Roadmap for Software Leaders**

**Extreme Programming in Practice**

**Let Over Lambda**

**Another day without Test-Driven Development means more time wasted chasing bugs and watching your code deteriorate. You thought TDD was for someone else, but it's not! It's for you, the embedded C programmer. TDD helps you prevent defects and build software with a long useful life. This is the first book to teach the hows and whys of TDD for C programmers. TDD is a modern programming practice C developers need to know. It's a different way to program---unit tests are written in a tight feedback loop with the production code, assuring your code does what you think. You get valuable feedback every few minutes. You find mistakes before they become bugs. You get early warning of design problems. You get immediate notification of side effect defects. You get to spend more time adding valuable features to your product. James is one of the few experts in applying TDD to embedded C. With his 1.5 decades of training, coaching, and practicing TDD in C, C++, Java, and C# he will lead you from being a novice in TDD to using the techniques that few have mastered. This book is full of code written for embedded C programmers. You don't just see the end product, you see code and tests evolve. James leads you through the thought process and decisions made each step of the way. You'll learn techniques for test-driving code right next to the hardware, and you'll learn design principles and how to apply them to C to keep your code clean and flexible. To run the examples in this book, you will need a C/C++ development environment on your machine, and the GNU GCC tool chain or Microsoft Visual Studio for C++ (some project conversion may be needed).**

**The latest title in Addison Wesley's world-renowned Robert C. Martin Series on better software development, Code That Fits in Your Head offers indispensable practical advice for writing code at a sustainable pace, and controlling the complexity that causes too many software projects to spin out of control. Reflecting decades of experience consulting on software projects and helping development teams succeed, Mark Seemann shares proven practices and heuristics, supported by realistic advice. His guidance ranges from checklists to teamwork, encapsulation to decomposition, API design to unit testing and troubleshooting. Throughout, Seemann illuminates his insights with up-to-date code examples drawn from a start to finish sample project. Seemann's examples are written in C#, and designed to be clear and useful to every object-oriented enterprise developer, whether they use C#, Java, or another language. Code That Fits in Your Head is accompanied by the complete code base for this sample application, organized in a Git repository to facilitate further exploration of details that don't fit in the text.**

**Let Over Lambda is one of the most hardcore computer programming books out there. Starting with the fundamentals, it describes the most advanced features of the most advanced language: Common Lisp. Only the top percentile of programmers use lisp and if you can understand this book you are in the top percentile of lisp programmers. If you are looking for a dry coding manual that re-hashes common-sense techniques in whatever langue du jour, this book is not for you. This book is about pushing the boundaries of what we know about programming. While this book teaches useful skills that can help solve your programming problems today and now, it has also been designed to be entertaining and inspiring. If you have ever wondered what lisp or even programming itself is really about, this is the book you have been looking for.**

**With the award-winning book Agile Software Development: Principles, Patterns, and Practices, Robert C. Martin helped bring Agile principles to tens of thousands of Java and C++ programmers. Now .NET programmers have a definitive guide to agile methods with this completely updated volume from Robert C. Martin and Micah Martin, Agile Principles, Patterns, and Practices in C#. This book presents a series of case studies illustrating the fundamentals of Agile development and Agile design, and moves quickly from UML models to real C# code. The**

*introductory chapters lay out the basics of the agile movement, while the later chapters show proven techniques in action. The book includes many source code examples that are also available for download from the authors' Web site. Readers will come away from this book understanding Agile principles, and the fourteen practices of Extreme Programming Spiking, splitting, velocity, and planning iterations and releases Test-driven development, test-first design, and acceptance testing Refactoring with unit testing Pair programming Agile design and design smells The five types of UML diagrams and how to use them effectively Object-oriented package design and design patterns How to put all of it together for a real-world project Whether you are a C# programmer or a Visual Basic or Java programmer Learning C#, a software development manager, or a business analyst, Agile Principles, Patterns, and Practices in C# is the first book you should read to understand agile software and how it applies to programming in the .NET Framework.*

**Support Constant Change**

**Refactor Your Wetware**

**Principles, Patterns, and Practices**

**Code Complete**

**Clean Code**

**Asynchronous, Parallel, and Multithreaded Programming**

**Get Your Hands Dirty on Clean Architecture**

What's the best approach for developing an application with JavaScript? This book helps you answer that question with numerous JavaScript coding patterns and best practices. If you're an experienced developer looking to solve problems related to objects, functions, inheritance, and other language-specific categories, the abstractions and code templates in this guide are ideal—whether you're using JavaScript to write a client-side, server-side, or desktop application. Written by JavaScript expert Stoyan Stefanov—Senior Yahoo! Technical and architect of YSlow 2.0, the web page performance optimization tool—JavaScript Patterns includes practical advice for implementing each pattern discussed, along with several hands-on examples. You'll also learn about anti-patterns: common programming approaches that cause more problems than they solve. Explore useful habits for writing high-quality JavaScript code, such as avoiding globals, using single var declarations, and more Learn why literal notation patterns are simpler alternatives to constructor functions Discover different ways to define a function in JavaScript Create objects that go beyond the basic patterns of using object literals and constructor functions Learn the options available for code reuse and inheritance in JavaScript Study sample JavaScript approaches to common design patterns such as Singleton, Factory, Decorator, and more Examine patterns that apply specifically to the client-side browser environment

What others in the trenches say about The Pragmatic Programmer... “The cool thing about this book is that it’s great for keeping the programming process fresh. The book helps you to continue to grow and clearly comes from people who have been there.” —Kent Beck, author of Extreme Programming Explained: Embrace Change “I found this book to be a great mix of solid advice and wonderful analogies!” —Martin Fowler, author of Refactoring and UML Distilled “I would buy a copy, read it twice, then tell all my colleagues to run out and grab a copy. This is a book I would never loan because I would worry about it being lost.” —Kevin Ruland, Management Science, MSG-Logistics “The wisdom and practical experience of the authors is obvious. The topics presented are relevant and useful.... By far its greatest strength for me has been the outstanding analogies—tracer bullets, broken windows, and the fabulous helicopter-based explanation of the need for orthogonality, especially in a crisis situation. I have little doubt that this book will eventually become an excellent source of useful information for journeymen programmers and expert mentors alike.” —John Lakos, author of Large-Scale C++ Software Design “This is the sort of book I will buy a dozen copies of when it comes out so I can give it to my clients.” —Eric Vought, Software Engineer “Most modern books on software development fail to cover the basics of what makes a great software developer, instead spending their time on syntax or technology where in reality the greatest leverage possible for any software team is in having talented developers who really know their craft well. An excellent book.” —Pete McBreen, Independent Consultant “Since reading this book, I have implemented many of the practical suggestions and tips it contains. Across the board, they have saved my company time and money while helping me get my job done quicker! This should be a desktop reference for everyone who works with code for a living.” —Jared Richardson, Senior Software Developer, iRenaissance, Inc. “I would like to see this issued to every new employee at my company....” —Chris Cleeland, Senior Software Engineer, Object Computing, Inc. “If I’m putting together a project, it’s the authors of this book that I want. . . . And failing that I’d settle for people who’ve read their book.” —Ward Cunningham Straight from the programming trenches, The Pragmatic Programmer cuts through the increasing specialization and technicalities of modern software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to Fight software rot; Avoid the trap of duplicating knowledge; Write flexible, dynamic, and adaptable code; Avoid programming by coincidence; Bullet-proof your code with contracts, assertions, and exceptions; Capture real requirements; Test ruthlessly and effectively; Delight your users; Build teams of pragmatic programmers; and Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, The Pragmatic Programmer illustrates the best practices and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects, use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes

that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer.

More C++ Gems picks up where the first book left off, presenting tips, tricks, proven strategies, easy-to-follow techniques, and usable source code.

The Complete Adult Psychotherapy Treatment Planner, Fourth Edition provides all the elements necessary to quickly and easily develop formal treatment plans that satisfy the demands of HMOs, managed care companies, third-party payors, and state and federal agencies. New edition features: Empirically supported, evidence-based treatment interventions Organized around 43 main presenting problems, including anger management, chemical dependence, depression, financial stress, low self-esteem, and Obsessive-Compulsive Disorder Over 1,000 prewritten treatment goals, objectives, and interventions - plus space to record your own treatment plan options Easy-to-use reference format helps locate treatment plan components by behavioral problem Designed to correspond with the The Adult Psychotherapy Progress Notes Planner, Third Edition and the Adult Psychotherapy Homework Planner, Second Edition Includes a sample treatment plan that conforms to the requirements of most third-party payors and accrediting agencies (including CARF, JCAHO, and NCQA).

The Complete Adult Psychotherapy Treatment Planner

Code That Fits in Your Head

The Clean Coder

UML for Java Programmers

your journey to mastery, 20th Anniversary Edition

50 Years of Lisp

Release It!

*In Clean Craftmanship: Programming with Pride, the legendary Robert C. Martin ("Uncle Bob") has written every programmer's definitive guide to working well. Martin brings together the disciplines, standards, and ethics you need to deliver robust, effective code quickly and productively, and be proud of all the software you write - every single day. Martin, the best-selling author of The Clean Coder, begins with a pragmatic, technical, and prescriptive guide to five foundational disciplines of software craftsmanship: test-driven development, refactoring, simple design, collaborative programming (pairing), and acceptance tests. Next, he moves up to standards -- outlining the baseline expectations the world has of software developers, illuminating how those often differ from their own perspectives, and helping you repair the mismatch. Finally, he turns to the ethics of the programming profession, describing ten fundamental promises all software developers should make to their colleagues, their users, and above all, themselves. With Martin's guidance and advice, you can consistently write code that builds trust instead of undermining it: trust among your users, and throughout a society that depends on software for its very survival.*

*If you're one of the many developers uncertain about concurrent and multithreaded development, this practical cookbook will change your mind. With more than 75 code-rich recipes, author Stephen Cleary demonstrates parallel processing and asynchronous programming techniques, using libraries and language features in .NET 4.5 and C# 5.0. Concurrency is becoming more common in responsive and scalable application development, but it's been extremely difficult to code. The detailed solutions in this cookbook show you how modern tools raise the level of abstraction, making concurrency much easier than before. Complete with ready-to-use code and discussions about how and why the solution works, you get recipes for using: async and await for asynchronous operations Parallel programming with the Task Parallel Library The TPL Dataflow library for creating dataflow pipelines Capabilities that Reactive Extensions build on top of LINQ Unit testing with concurrent code Interop scenarios for combining concurrent approaches Immutable, threadsafe, and producer/consumer collections Cancellation support in your concurrent code Asynchronous-friendly Object-Oriented Programming Thread synchronization for accessing data*

*Practical Software Architecture Solutions from the Legendary Robert C. Martin ("Uncle Bob") By applying universal rules of software architecture, you can dramatically improve developer productivity throughout the life of any software system. Now, building upon the success of his best-selling books Clean Code and The Clean Coder, legendary software craftsman Robert C. Martin ("Uncle Bob") reveals those rules and helps you apply them. Martin's Clean Architecture doesn't merely present options. Drawing on over a half-century of experience in software environments of every imaginable type, Martin tells you what choices to make and why they are critical to your success. As you've come to expect from Uncle Bob, this book is packed with direct, no-nonsense solutions for the real challenges you'll face—the ones that will make or break your projects. Learn what software architects need to achieve—and core disciplines and practices for achieving it Master essential software design principles for addressing function, component separation, and data management See how programming paradigms impose discipline by restricting what developers can do Understand what's critically important and what's merely a "detail" Implement optimal, high-level structures for web, database, thick-client, console, and embedded applications Define appropriate boundaries and layers, and organize components and services See why designs and architectures go wrong, and how to prevent (or fix) these failures Clean Architecture is essential reading for every current or aspiring software architect, systems analyst, system designer, and software manager—and for every programmer who must execute someone else's designs. Register your product for convenient access to downloads, updates, and/or corrections as they become available.*

*The Unified Modeling Language has become the industry standard for the expression of software designs. The Java programming language continues to grow in popularity as the language of choice for the serious application developer. Using UML and Java together would appear to be a natural marriage, one that can produce considerable benefit. However, there are nuances that the seasoned developer needs to keep in mind when using UML and Java together. Software expert Robert Martin presents a concise guide, with numerous examples, that will help the programmer*

*leverage the power of both development concepts. The author ignores features of UML that do not apply to java programmers, saving the reader time and effort. He provides direct guidance and points the reader to real-world usage scenarios. The overall practical approach of this book brings key information related to Java to the many presentations. The result is an highly practical guide to using the UML with Java.*

*The Productive Programmer*

*How Google Tests Software*

*Creating a Remarkable Career in Software Development*

*Test Driven Development for Embedded C*

*Clean Craftmanship*

*Pragmatic Thinking and Learning*

*Agile Principles, Patterns, and Practices in C#*

Printed in full color. Software development happens in your head. Not in an editor, IDE, or designtool. You're well educated on how to work with software and hardware, but what about wetware--our own brains? Learning new skills and new technology is critical to your career, and it's all in your head. In this book by Andy Hunt, you'll learn how our brains are wired, and how to take advantage of your brain's architecture. You'll learn new tricks and tipsto learn more, faster, and retain more of what you learn. You need a pragmatic approach to thinking and learning. You need to Refactor Your Wetware.

Programmers have to learn constantly; not just the stereotypical new technologies, but also the problem domain of the application, the whims of the user community, the quirks of your teammates, the shifting sands of the industry, and the evolving characteristics of the project itself as it is built. We'll journey together through bits of cognitive and neuroscience, learning and behavioral theory. You'll see some surprising aspects of how our brains work, and how you can take advantage of the system to improve your own learning and thinking skills. In this book you'll learn how to: Use the Dreyfus Model of Skill Acquisition to become more expert Leverage the architecture of the brain to strengthen different thinking modes Avoid common "known bugs" in your mind Learn more deliberately and more effectively Manage knowledge more efficiently

Winner of the 2011 Jolt Excellence Award! Getting software released to users is often a painful, risky, and time-consuming process. This groundbreaking new book sets out the principles and technical practices that enable rapid, incremental delivery of high quality, valuable new functionality to users. Through automation of the build, deployment, and testing process, and improved collaboration between developers, testers, and operations, delivery teams can get changes released in a matter of hours— sometimes even minutes—no matter what the size of a project or the complexity of its code base. Jez Humble and David Farley begin by presenting the foundations of a rapid, reliable, low-risk delivery process. Next, they introduce the “deployment pipeline,” an automated process for managing all changes, from check-in to release. Finally, they discuss the “ecosystem” needed to support continuous delivery, from infrastructure, data and configuration management to governance. The authors introduce state-of-the-art techniques, including automated infrastructure management and data migration, and the use of virtualization. For each, they review key issues, identify best practices, and demonstrate how to mitigate risks. Coverage includes • Automating all facets of building, integrating, testing, and deploying software • Implementing deployment pipelines at team and organizational levels • Improving collaboration between developers, testers, and operations • Developing features incrementally on large and distributed teams • Implementing an effective configuration management strategy • Automating acceptance testing, from analysis to implementation • Testing capacity and other non-functional requirements • Implementing continuous deployment and zero-downtime releases • Managing infrastructure, data, components and dependencies • Navigating risk management, compliance, and auditing Whether you're a developer, systems administrator, tester, or manager, this book will help your organization move from idea to release faster than ever—so you can deliver value to your business rapidly and reliably. This comprehensive, pragmatic tutorial on Agile Development and eXtreme programming, written by one of the founding fathers of Agile Development: 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.

This title focuses on the most critical aspects of software development: building robust, bug free systems, meeting deadlines, and coming in under budget. It includes artifacts, anecdotes, and actual code from an enterprise-class XP project.

Working Effectively with Legacy Code

Elegant Objects

Build Better Applications with Coding and Design Patterns

AGILE PRIN PATTS PRACTS C#\_1

Dreaming in Code

More C++ Gems

The Pragmatic Programmer

**Anyone who develops software for a living needs a proven way to produce it better, faster, and cheaper. The Productive Programmer offers critical timesaving and productivity tools that you can adopt right away, no matter what platform you use. Master developer Neal Ford not only offers advice on the mechanics of productivity—how to work smarter, spurn interruptions, get the most out your computer, and avoid repetition—he also details valuable practices that will help you elude common traps, improve your code, and become more valuable to your team. You'll learn to: Write the test before you write the code Manage the lifecycle of your objects fastidiously Build only what you need now, not what you might need later Apply ancient philosophies to software development Question authority, rather than blindly adhere to standards Make hard things easier and impossible things possible through meta-programming Be sure all code within a method is at the same level of abstraction Pick the right editor and assemble the best tools for the job This isn't theory, but the fruits of Ford's real-world experience as an Application Architect at the global IT consultancy ThoughtWorks. Whether you're a beginner or a pro with years of experience, you'll improve your work and your career with the simple and straightforward principles in The Productive Programmer.**

In Clean Craftmanship , the legendary Robert C. Martin ("Uncle Bob") has written every programmer's definitive guide to working well. Martin brings together the disciplines, standards, and ethics you need to deliver robust, effective code quickly and productively, and be proud of all the software you write -- every single day. Martin, the

best-selling author of *The Clean Coder*, begins with a pragmatic, technical, and prescriptive guide to five foundational disciplines of software craftsmanship: test-driven development, refactoring, simple design, collaborative programming (pairing), and acceptance tests. Next, he moves up to standards -- outlining the baseline expectations the world has of software developers, illuminating how those often differ from their own perspectives, and helping you repair the mismatch. Finally, he turns to the ethics of the programming profession, describing ten fundamental promises all software developers should make to their colleagues, their users, and above all, themselves. With Martin's guidance and advice, you can consistently write code that builds trust instead of undermining it -- trust among your users and throughout a society that depends on software for its very survival.

You want increased customer satisfaction, faster development cycles, and less wasted work. Domain-driven design (DDD) combined with functional programming is the innovative combo that will get you there. In this pragmatic, down-to-earth guide, you'll see how applying the core principles of functional programming can result in software designs that model real-world requirements both elegantly and concisely - often more so than an object-oriented approach. Practical examples in the open-source F# functional language, and examples from familiar business domains, show you how to apply these techniques to build software that is business-focused, flexible, and high quality. Domain-driven design is a well-established approach to designing software that ensures that domain experts and developers work together effectively to create high-quality software. This book is the first to combine DDD with techniques from statically typed functional programming. This book is perfect for newcomers to DDD or functional programming - all the techniques you need will be introduced and explained. Model a complex domain accurately using the F# type system, creating compilable code that is also readable documentation---ensuring that the code and design never get out of sync. Encode business rules in the design so that you have "compile-time unit tests," and eliminate many potential bugs by making illegal states unrepresentable. Assemble a series of small, testable functions into a complete use case, and compose these individual scenarios into a large-scale design. Discover why the combination of functional programming and DDD leads naturally to service-oriented and hexagonal architectures. Finally, create a functional domain model that works with traditional databases, NoSQL, and event stores, and safely expose your domain via a website or API. Solve real problems by focusing on real-world requirements for your software. What You Need: The code in this book is designed to be run interactively on Windows, Mac and Linux. You will need a recent version of F# (4.0 or greater), and the appropriate .NET runtime for your platform. Full installation instructions for all platforms at [fsharp.org](http://fsharp.org).

Get the most out of JavaScript for building web applications through a series of patterns, techniques, and case studies for clean coding Key Features Write maintainable JS code using internal abstraction, well-written tests, and well-documented code Understand the agents of clean coding like SOLID principles, OOP, and functional programming Explore solutions to tackle common JavaScript challenges in building UIs, managing APIs, and writing states Book Description Building robust apps starts with creating clean code. In this book, you'll explore techniques for doing this by learning everything from the basics of JavaScript through to the practices of clean code. You'll write functional, intuitive, and maintainable code while also understanding how your code affects the end user and the wider community. The book starts with popular clean-coding principles such as SOLID, and the Law of Demeter (LoD), along with highlighting the enemies of writing clean code such as cargo culting and over-management. You'll then delve into JavaScript, understanding the more complex aspects of the language. Next, you'll create meaningful abstractions using design patterns, such as the Class Pattern and the Revealing Module Pattern. You'll explore real-world challenges such as DOM reconciliation, state management, dependency management, and security, both within browser and server environments. Later, you'll cover tooling and testing methodologies and the importance of documenting code. Finally, the book will focus on advocacy and good communication for improving code cleanliness within teams or workplaces, along with covering a case study for clean coding. By the end of this book, you'll be well-versed with JavaScript and have learned how to create clean abstractions, test them, and communicate about them via documentation. What you will learn Understand the true purpose of code and the problems it solves for your end-users and colleagues Discover the tenets and enemies of clean code considering the effects of cultural and syntactic conventions Use modern JavaScript syntax and design patterns to craft intuitive abstractions Maintain code quality within your team via wise adoption of tooling and advocating best practices Learn the modern ecosystem of

JavaScript and its challenges like DOM reconciliation and state management Express the behavior of your code both within tests and via various forms of documentation Who this book is for This book is for anyone who writes JavaScript, professionally or otherwise. As this book does not relate specifically to any particular framework or environment, no prior experience of any JavaScript web framework is required. Some knowledge of programming is assumed to understand the concepts covered in the book more effectively.

Clean Craftsmanship

Disciplines, Standards, and Ethics

JavaScript Patterns

Patterns, Principles, and Practices of Domain-Driven Design

Heuristics for Software Engineering

The Robert C. Martin Clean Code Collection (Collection)

A Code of Conduct for Professional Programmers

Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.

In this comprehensive yet accessible overview for software leaders, the author presents an impactful, action-oriented prescription—covering the practical considerations needed to ensure you reap the full benefits of effective Agile

2012 Jolt Award finalist! Pioneering the Future of Software Test Do you need to get it right, too? Then, learn from Google. Legendary testing expert James Whittaker, until recently a Google testing leader, and two top Google experts reveal exactly how Google tests software, offering brand-new best practices you can use even if you're not quite Google's size...yet! Breakthrough Techniques You Can Actually Use Discover 100% practical, amazingly scalable techniques for analyzing risk and planning tests...thinking like real users...implementing exploratory, black box, white box, and acceptance testing...getting usable feedback...tracking issues...choosing and creating tools...testing "Docs & Mocks," interfaces, classes, modules, libraries, binaries, services, and infrastructure...reviewing code and refactoring...using test hooks, presubmit scripts, queues, continuous builds, and more. With these techniques, you can transform testing from a bottleneck into an accelerator—and make your whole organization more productive!

The software development ecosystem is constantly changing, providing a constant stream of new tools, frameworks, techniques, and paradigms. Over the past few years, incremental developments in core engineering practices for software development have created the foundations for rethinking how architecture changes over time, along with ways to protect important architectural characteristics as it evolves. This practical guide ties those parts together with a new way to think about architecture and time.

Design and Deploy Production-Ready Software

A hands-on guide to creating clean web applications with code examples in Java

Building Evolutionary Architectures

Back to Basics

Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)

Clean Code in JavaScript

Gain insight into how hexagonal architecture can help to keep the cost of development low over the complete lifetime of an application Key Features Explore ways to make your software flexible, extensible, and adaptable Learn new concepts that you can easily blend with your own software development style Develop the mindset of building maintainable solutions instead of taking shortcuts Book Description We would all like to build software architecture that yields adaptable and flexible software with low development costs. But, unreasonable deadlines and shortcuts make it very hard to create such an architecture. Get Your Hands Dirty on Clean Architecture starts with a discussion about the conventional layered architecture style and its disadvantages. It also talks about the advantages of the domain-centric architecture styles of Robert C. Martin's Clean Architecture and Alistair Cockburn's Hexagonal Architecture. Then, the book dives into hands-on chapters that show you how to manifest a hexagonal architecture in actual code. You'll learn in detail about different mapping strategies between the layers of a hexagonal architecture and see how to assemble the architecture elements into an application. The later chapters demonstrate how to enforce architecture boundaries. You'll also learn what shortcuts produce what types of technical debt and how, sometimes, it is a good idea to willingly take on those debts. After reading this book, you'll have all the knowledge you need to create applications using the hexagonal architecture style of web development. What you will learn Identify potential shortcomings of using a layered architecture Apply methods to enforce architecture boundaries Find out how potential shortcuts can affect the software architecture Produce arguments for when to use which style of architecture Structure your code according to the architecture Apply various types of tests that will cover each element of the architecture Who this book is for This book is for you if you care about the architecture of the software you are building. To get the most out of this book, you must have some experience with web development. The code examples in this book are in Java. If you are not a Java programmer but can read object-oriented code in other languages, you will be fine. In the few places where Java or framework specifics are needed, they are thoroughly explained.

Clean Code A Handbook of Agile Software Craftsmanship Pearson Education

Our civilization runs on software. Yet the art of creating it continues to be a dark mystery, even to the experts. To find out

why it's so hard to bend computers to our will, Scott Rosenberg spent three years following a team of maverick software developers—led by Lotus 1-2-3 creator Mitch Kapor—designing a novel personal information manager meant to challenge market leader Microsoft Outlook. Their story takes us through a maze of abrupt dead ends and exhilarating breakthroughs as they wrestle not only with the abstraction of code, but with the unpredictability of human behavior—especially their own.

TL;DR Compound variable names, validators, private static literals, configurable objects, inheritance, annotations, MVC, dependency injection containers, reflection, ORM and even algorithms are our enemies.

Develop reliable, maintainable, and robust JavaScript

Concurrency in C# Cookbook

Professionalism, Pragmatism, Pride

Continuous Delivery

Clean Agile

Agile Software Development

WORK EFFECT LEG CODE p1

Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of heuristics and "smells" accumulated from the process of writing clean code.

"One of the most significant books in my life." –Obie Fernandez, Author, The Rails Way "Twenty years ago, the first edition of The Pragmatic Programmer completely changed the trajectory of my career. This new edition could do the same for yours." –Mike Cohn, Author of Succeeding with Agile, Agile Estimating and Planning, and User Stories Applied ". . . filled with practical advice, both technical and professional, that will serve you and your projects well for years to come." –Andrea Goulet, CEO, Corgibytes, Founder, LegacyCode.Rocks ". . . lightning does strike twice, and this book is proof." –VM (Vicky) Brasseur, Director of Open Source Strategy, Juniper Networks The Pragmatic Programmer is one of those rare tech books you'll read, re-read, and read again over the years. Whether you're new to the field or an experienced practitioner, you'll come away with fresh insights each and every time. Dave Thomas and Andy Hunt wrote the first edition of this influential book in 1999 to help their clients create better software and rediscover the joy of coding. These lessons have helped a generation of programmers examine the very essence of software development, independent of any particular language, framework, or methodology, and the Pragmatic philosophy has spawned hundreds of books, screencasts, and audio books, as well as thousands of careers and success stories. Now, twenty years later, this new edition re-examines what it means to be a modern programmer. Topics range from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to: Fight software rot Learn continuously Avoid the trap of duplicating knowledge Write flexible, dynamic, and adaptable code Harness the power of basic tools Avoid programming by coincidence Learn real requirements Solve the underlying problems of concurrent code Guard against security vulnerabilities Build teams of Pragmatic Programmers Take responsibility for your work and career Test ruthlessly and effectively, including property-based testing Implement the Pragmatic Starter Kit Delight your users Written as a series of self-contained sections and filled with classic and fresh anecdotes, thoughtful examples, and interesting analogies, The Pragmatic Programmer illustrates the best approaches and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects, use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Agile Values and Principles for a New Generation "In the journey to all things Agile, Uncle Bob has been there, done that, and has the both the t-shirt and the scars to show for it. This delightful book is part history, part personal stories, and all wisdom. If you want to understand what Agile is and how it came to be, this is the book for you." –Grady Booch "Bob's frustration colors every sentence of Clean Agile, but it's a justified frustration. What is in the world of Agile development is nothing compared to what could be. This book is Bob's perspective on what to focus on to get to that 'what could be.' And he's been there, so it's worth listening." –Kent Beck "It's good to read Uncle Bob's take on Agile. Whether just beginning, or a seasoned Agilista, you would do well to read this book. I agree with almost all of it. It's just some of the parts make me realize my own shortcomings, dammit. It made me double-check our code coverage (85.09%)." –Jon Kern Nearly twenty years after the Agile Manifesto was first presented, the legendary Robert C. Martin ("Uncle Bob") reintroduces Agile values and principles for a new generation—programmers and nonprogrammers alike. Martin, author of Clean Code and other highly influential software development guides, was there at Agile's founding. Now, in Clean Agile: Back to Basics, he strips away misunderstandings and distractions that over the years have made it harder to use Agile than was originally intended. Martin describes what Agile is in no uncertain terms: a small discipline that helps small teams

manage small projects . . . with huge implications because every big project is comprised of many small projects. Drawing on his fifty years' experience with projects of every conceivable type, he shows how Agile can help you bring true professionalism to software development. Get back to the basics—what Agile is, was, and should always be. Understand the origins, and proper practice, of SCRUM Master essential business-facing Agile practices, from small releases and acceptance tests to whole-team communication. Explore Agile team members' relationships with each other, and with their product. Rediscover indispensable Agile technical practices: TDD, refactoring, simple design, and pair programming. Understand the central roles values and craftsmanship play in your Agile team's success. If you want Agile's true benefits, there are no shortcuts: You need to do Agile right. *Clean Agile: Back to Basics* will show you how, whether you're a developer, tester, manager, project manager, or customer. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

In *The Software Craftsman*, Sandro Mancuso explains what craftsmanship means to the developer and his or her organization, and shows how to live it every day in your real-world development environment. Mancuso shows how software craftsmanship fits with and helps students improve upon best-practice technical disciplines such as agile and lean, taking all development projects to the next level. Readers will learn how to change the disastrous perception that software developers are the same as factory workers, and that software projects can be run like factories.

From Journeyman to Master