Download Principle Based Refactoring Learning Software Design Principles By Applying Refactoring Rules

Recognizing the pretension ways to get this ebook principle based refactoring learning software design principles by applying refactoring rules is additionally useful. You have remained in right site to begin getting this info. get the principle based refactoring learning software design principles by applying refactoring rules associate that we meet the expense of here and check out the link.

You could purchase guide principle based refactoring learning software design principles by applying refactoring rules or get it as soon as feasible. You could quickly download this principle based refactoring learning software design principles by applying refactoring rules after getting deal. So, later than you require the book swiftly, you can straight acquire it. Its fittingly utterly easy and suitably fats, isnt it? You have to favor to in this aerate

Principle-Based Refactoring  - Steve Halladay  - 2012-08-31
You know good software when you see it, but how do you explain what good software is? Experienced software developers have pet practices and techniques that make their software easier to test, maintain and understand. But when you ask them how to make your software like theirs, they give you a seemingly endless list of rules. How can they remember all those rules? The secret is that they dont! Instead, experienced software developers understand a handful of basic principles. The rules are merely manifestations of these basic principles. But, principles are hard to explain; so experienced developers resort to explaining rules instead. In Principle-Based Refactoring, Halladay explains a set of software refactoring rules and links the refactoring rules back to the basic principles that drive robust software design. The book identifies eight fundamental design principles and also includes a set of approximately fifty refactoring rules that illustrate the principles. Each rule has a summary description, a discussion, including references back to the driving principles, and examples of the rules applications. In addition, this book discusses refactoring mechanics including test strategies that guide software developers in verifying the quality of refactored code.

Re-Engineering Legacy Software  - Chris Birchall  - 2016-04-15
Summary As a developer, you may inherit projects built on existing codebases with design patterns, usage assumptions, infrastructure, and tooling from another time and another team. Fortunately, there are ways to breathe new life into legacy projects so you can maintain, improve, and scale them without fighting their limitations. Purchase of the print book includes a free ebook in PDF, Kindle, and ePUB formats from Manning Publications. About the Book Re-Engineering Legacy Software is an experience-driven guide to revitalizing inherited projects. It covers refactoring, quality metrics, toolchain and workflow, continuous integration, infrastructure automation, and organizational culture. You'll learn techniques for introducing dependency injection for code modularity, quantitatively measuring quality, and automating infrastructure. You'll also develop practical processes for deciding whether to rewrite or refactor, organizing teams, and convincing management that quality matters. Core topics include deciphering and modularizing awkward code structures, integrating and automating tests, replacing outdated build systems, and using tools like Vagrant and Ansible for infrastructure automation. What's Inside Refactoring legacy codebases Continuous inspection and integration Automating legacy infrastructure New tests for old code Modularizing monolithic projects About the Reader This book is written for developers and team leads comfortable with an OO language like Java or C#. About the Author Chris Birchall is a senior developer at the Guardian in London, working on the back-end services that power the website. Table of Contents PART 1 GETTING STARTED Understanding the challenges of legacy projects Finding your starting point PART 2 REFACTORING TO IMPROVE THE CODEBASE Preparing to refactor Refactoring Re-architecting The Big

Future-ProOF Software-Systems  - Frank J. Furrer  - 2019-09-25
This book focuses on software architecture and the value of architecture in the development of long-lived, mission-critical, trustworthy software-systems. The author introduces and demonstrates the powerful strategy of "Managed Evolution," along with the engineering best practice known as "Principle-based Architecting." The book examines in detail architecture principles for e.g., Business Value, Changeability, Resilience, and Dependability. The author argues that the software development community has a strong responsibility to produce and operate useful, dependable, and trustworthy software. Software should at the same time provide business value and guarantee many quality-of-service properties, including security, safety, performance, and integrity. As Dr. Furrer states, "Producing dependable software is a balancing act between investing in the implementation of business functionality and investing in the quality-of-service properties of the software-systems." The book presents extensive coverage of such concepts as: Principle-Based Architecting Managed Evolution Strategy The Future Principles for Business Value Legacy Software Modernization/Migration Architecture Principles for Changeability Architecture Principles for Resilience Architecture Principles for Dependability The text is supplemented with numerous figures, tables, examples and illustrative quotations. Future-Proof Software-Systems provides a set of good engineering practices, devised for integration into most software development processes dedicated to the creation of software-systems that incorporate Managed Evolution.

Principles for Business Value Legacy Software Modernization/Migration Architecture Principles for Changeability

Architecture Principles for Resilience Architecture Principles for Dependability The text is supplemented with numerous figures, tables, examples and illustrative quotations. Future-Proof Software-Systems provides a set of good engineering practices, devised for integration into most software development processes dedicated to the creation of software-systems that incorporate Managed Evolution.

Future-ProOF Software-Systems  - Frank J. Furrer  - 2019-09-25
This book focuses on software architecture and the value of architecture in the development of long-lived, mission-critical, trustworthy software-systems. The author introduces and demonstrates the powerful strategy of "Managed Evolution," along with the engineering best practice known as "Principle-based Architecting." The book examines in detail architecture principles for e.g., Business Value, Changeability, Resilience, and Dependability. The author argues that the software development community has a strong responsibility to produce and operate useful, dependable, and trustworthy software. Software should at the same time provide business value and guarantee many quality-of-service properties, including security, safety, performance, and integrity. As Dr. Furrer states, "Producing dependable software is a balancing act between investing in the implementation of business functionality and investing in the quality-of-service properties of the software-systems." The book presents extensive coverage of such concepts as: Principle-Based Architecting Managed Evolution Strategy The Future
Agile Technical Practices Distilled

Pedro M. Santos - 2019-06-28

Delve deep into the various technical practices, principles, and values of Agile. Key Features Discover the essence of Agile software development and the key principles of software design. Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration. Learn and apply the four elements of simple design. Book Description The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. The book will help you discover new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn: Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more. Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master Use code smells as feedback to improve your design. Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests. Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related. Improve the understanding of your business domain using BDD and other principles for “doing the right thing, not only the thing right!” Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner’s book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

Agile Technical Practices Distilled

Pedro M. Santos - 2019-06-28

Delve deep into the various technical practices, principles, and values of Agile. Key Features Discover the essence of Agile software development and the key principles of software design. Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration. Learn and apply the four elements of simple design. Book Description The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. The book will help you discover new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn: Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more. Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master Use code smells as feedback to improve your design. Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests. Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related. Improve the understanding of your business domain using BDD and other principles for “doing the right thing, not only the thing right!” Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner’s book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

Agile Technical Practices Distilled

Pedro M. Santos - 2019-06-28

Delve deep into the various technical practices, principles, and values of Agile. Key Features Discover the essence of Agile software development and the key principles of software design. Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration. Learn and apply the four elements of simple design. Book Description The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. The book will help you discover new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn: Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more. Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master Use code smells as feedback to improve your design. Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests. Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related. Improve the understanding of your business domain using BDD and other principles for “doing the right thing, not only the thing right!” Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner’s book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

Agile Technical Practices Distilled

Pedro M. Santos - 2019-06-28

Delve deep into the various technical practices, principles, and values of Agile. Key Features Discover the essence of Agile software development and the key principles of software design. Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration. Learn and apply the four elements of simple design. Book Description The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. The book will help you discover new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn: Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more. Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master Use code smells as feedback to improve your design. Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests. Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related. Improve the understanding of your business domain using BDD and other principles for “doing the right thing, not only the thing right!” Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner’s book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.
principle-based-refactoring-learning-software-design-principles-by-applying-refactoring-rules

This book constitutes the refereed proceedings of the 4th International Conference on Fundamental Approaches Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration. Learn and apply the four elements of simple design book fundamentals. The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices can help you become a better programmer. This book uses the term Agile as a wide umbrella and covers Agile principles and practices, as well as most methodologies associated with it. You'll begin by discovering how driver-navigator, chess clock, and other techniques used in the pair programming approach introduce discipline while writing code. You'll then learn to safely change the design of your code using refactoring. While learning these techniques, you'll also explore various best practices to write efficient tests. The concluding chapters of the book delve deep into the SOLID principles - the five design principles that you can use to make your software more understandable, flexible and maintainable. By the end of the book, you will have discovered new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn Apply the red, green, refactor cycle of TDD to solve procedural problems Implement the various techniques used in the pair programming approach Use code smells as feedback Test your production code using mocks and stubs Refactor legacy code to bring it in line with modern Agile standards Apply the object calisthenics ruleset to enhance your software design Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner's book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

Information Systems Development - George Angelos Papadopoulos - 2009-09-23

This volume constitutes the published proceedings of the 17th International Conference on Information Systems Development. They present the latest and greatest concepts, approaches, and techniques of systems development - a notoriously transitional field.

Information Systems Development - George Angelos Papadopoulos - 2009-09-23

This volume constitutes the published proceedings of the 17th International Conference on Information Systems Development. They present the latest and greatest concepts, approaches, and techniques of systems development - a notoriously transitional field.


Issues in Software Research, Design, and Application: 2011 Edition is a ScholarlyEditions™ eBook that delivers timely, authoritative, and comprehensive information about Software Research, Design, and Application. The editors have built Issues in Software Research, Design, and Application: 2011 Edition on the vast information databases of ScholarlyNews™. You can expect the information about Software Research, Design, and Application in this eBook to be deeper than what you can access anywhere else, as well as consistently reliable, authoritative, informed, and relevant. The content of Issues in Software Research, Design, and Application: 2011 Edition has been produced by the world's leading scientists, engineers, analysts, research institutions, and companies. All of the content is from peer-reviewed sources, and all of it is written, edited, and compiled by the editors at ScholarlyEditions™ and available exclusively from us. You now have a source you can cite with authority, confidence, and credibility. More information is available at http://www.ScholarlyEditions.com/.


Issues in Software Research, Design, and Application: 2011 Edition is a ScholarlyEditions™ eBook that delivers timely, authoritative, and comprehensive information about Software Research, Design, and Application. The editors have built Issues in Software Research, Design, and Application: 2011 Edition on the vast information databases of ScholarlyNews™. You can expect the information about Software Research, Design, and Application in this eBook to be deeper than what you can access anywhere else, as well as consistently reliable, authoritative, informed, and relevant. The content of Issues in Software Research, Design, and Application: 2011 Edition has been produced by the world's leading scientists, engineers, analysts, research institutions, and companies. All of the content is from peer-reviewed sources, and all of it is written, edited, and compiled by the editors at ScholarlyEditions™ and available exclusively from us. You now have a source you can cite with authority, confidence, and credibility. More information is available at http://www.ScholarlyEditions.com/.

Fundamental Approaches to Software Engineering - Juan de Lara - 2012-03-09

This book constitutes the refereed proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering, FASE 2001, held in Genova, Italy in April 2001. The 22 revised full papers presented were carefully reviewed and selected from a total of 74 submissions. The papers are organized in topical sections on metamodeling, distributed components, UML testing, formal methods, and case studies.

Fundamental Approaches to Software Engineering - Juan de Lara - 2012-03-09

This book constitutes the refereed proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering, FASE 2001, held in Genova, Italy in April 2001. The 22 revised full papers presented were carefully reviewed and selected from a total of 74 submissions. The papers are organized in topical sections on metamodeling, distributed components, UML testing, formal methods, and case studies.


"This book presents cutting-edge research and analysis of the most recent advancements in the fields of database systems and software development"--Provided by publisher.


"This book presents cutting-edge research and analysis of the most recent advancements in the fields of database systems and software development"--Provided by publisher.

Advances in Computers -- 2014-01-11

Since its first volume in 1960, Advances in Computers has presented detailed coverage of innovations in computer hardware, software, theory, design, and applications. It has also provided contributors with a medium in which they can explore their subjects in greater depth and breadth than journal articles usually allow. As a result, many articles have become standard references that continue to be of significant, lasting value in this rapidly expanding field. In-depth surveys and tutorials on new computer technology Well-known authors and researchers in the field Extensive bibliographies with most chapters Many of the volumes are devoted to single themes or subfields of computer science.

Advances in Computers -- 2014-01-11

Since its first volume in 1960, Advances in Computers has presented detailed coverage of innovations in computer hardware, software, theory, design, and applications. It has also provided contributors with a medium in which they can explore their subjects in greater depth and breadth than journal articles usually allow. As a result, many articles have become standard references that continue to be of significant, lasting value in this rapidly expanding field. In-depth surveys and tutorials on new computer technology Well-known authors and researchers in the field Extensive bibliographies with most chapters Many of the volumes are devoted to single themes or subfields of computer science.

Computer Supported Education - H. Chad Lane - 2020-12-11

This book constitutes the thoroughly refereed proceedings of the 11th International Conference on Computer Supported Education, CSEDU 2019, held in Heraklion, Crete, Greece, in May 2019. The 30 revised full papers presented were carefully reviewed and selected from 202 submissions. The papers cover wide research fields including authoring tools and content development, AV-communication and multimedia, classroom management, e-Learning hardware and software, blended learning, critical success factors in distance learning.

Computer Supported Education - H. Chad Lane - 2020-12-11

This book constitutes the thoroughly refereed proceedings of the 11th International Conference on Computer Supported Education, CSEDU 2019, held in Heraklion, Crete, Greece, in May 2019. The 30 revised full papers presented were carefully reviewed and selected from 202 submissions. The papers cover wide research fields including authoring tools and content development, AV-communication and multimedia, classroom management, e-Learning hardware and software, blended learning, critical success factors in distance learning.

Software Quality, Complexity and Challenges of Software Engineering in Emerging Technologies - Dietmar Winkler - 2017-01-05

Software Quality, Complexity and Challenges of Software Engineering in Emerging Technologies - Dietmar Winkler - 2017-01-05
present the agile approach in a holistic and comprehensible learning environment that fits both industry and academia and inspires the spirit of agile software development. Agile software engineering is reviewed in this book through the following three perspectives: 1 The Human perspective, which includes cognitive and social aspects, and refers to learning and interpersonal processes between teammates, customers, and management. 1 The Organizational perspective, which includes managerial and cultural aspects, and refers to software project management and control. 1 The Technological perspective, which includes practical and technical aspects, and refers to design, testing, and coding, as well as to integration, delivery, and maintenance of software products. Specifically, we explain and analyze how the explicit attention that agile software development gives these perspectives and their interconnections, helps viii Preface it cope with the challenges of software projects. This multifaceted perspective on software development processes is reflected in this book, among other ways, by the chapter titles, which specify dimensions of software development projects such as quality, time, abstraction, and management, rather than specific project stages, phases, or practices.

Working Effectively with Legacy Code - Michael Feathers - 2004-09-22
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, and 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.

Working Effectively with Legacy Code - Michael Feathers - 2004-09-22
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, and 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.

Refactoring for Software Design Smells - Girish Suryanarayana - 2014-11-11
Awareness of design smells – indicators of common design problems – helps developers or software engineers understand mistakes made while designing, what design principles were overlooked or misapplied, and what principles need to be applied properly to address those smells through refactoring. Developers and software engineers may "know" principles and patterns, but are not aware of the "smells" that exist in their design because of wrong or misapplication of principles or patterns. These smells tend to contribute heavily to technical debt – engineers may "know" principles and patterns, but are not aware of the "smells" that exist in their design because of wrong or misapplication of principles or patterns. These smells tend to contribute heavily to technical debt – and need to be addressed via proper refactoring. Refactoring for Software Design Smells presents 25 structural design smells, their role in identifying design issues, and potential refactoring solutions. Organized across common areas of software design, each smell is presented with diagrams and examples illustrating the poor design practices and the problems that result, a catalog of nuggets of readily usable information that developers or engineers can apply in their projects. The authors distill their research and experience as consultants and trainers, providing insights that...
You’re a senior developer or a budding architect looking to understand an agile architect’s role by embracing agile architecture strategies and a lean-agile mindset. To understand the concepts covered in this book easily, you need to have prior knowledge of basic agile development practices.

**Becoming an Agile Software Architect**

*Rajesh R V - 2021-03-19*

A guide to successfully operating in a lean-agile organization for solutions architects and enterprise architects Key Features Develop the skills and processes to transform your organization to achieve organization-wide agility Who this book is for This agile study guide is for architects currently working on agile development projects or aspiring to work on agile software delivery, irrespective of the methodology they are using. You will also find this book useful if you’re looking to understand an agile architect’s role by embracing agile architecture strategies and a lean-agile mindset. To understand the concepts covered in this book easily, you need to have prior knowledge of basic agile development practices.
Five Lines of Code teaches refactoring that’s focused on concrete rules and getting any method down to five lines
Refactoring - Christian Clausen - 2021-11-09
As the application of object technology—particularly the Java programming language—has become commonplace, a new problem has emerged to confront the software development community. Significant numbers of poorly designed programs have been created by less-experienced developers, resulting in applications that are inefficient and hard to maintain and extend. Increasingly, software system professionals are discovering just how difficult it is to work with these inherited, “non-optimal” applications. For several years, expert-level object programmers have employed a growing collection of techniques to improve the structural integrity and functional performance of such existing software programs. Referred to as “refactoring,” these practices have remained in the domain of experts because no attempt has been made to transcribe the lore into a form that all developers could use... until now. In Refactoring: Improving the Design of Existing Code, renowned object technology mentor Martin Fowler breaks new ground, demystifying these master practices and demonstrating how software practitioners can realize the significant benefits of this new process. With proper training, a skilled system designer can take a bad design and rework it into well-designed, robust code. In this book, Martin Fowler shows you where opportunities for refactoring typically can be found, and how to go about reworking a bad design into a good one. Each refactoring step is simple—seemingly too simple to be worth doing. Refactoring may involve moving a field from one class to another, or pulling some code out of a method to turn it into its own method, or even pushing some code up or down a hierarchy. While these individual steps may seem elementary, the cumulative effect of such small changes can radically improve the design. Refactoring is a proven way to prevent software decay. In addition to discussing the various techniques of refactoring, the author provides a detailed catalog of more than seventy proven refactorings with helpful pointers that teach you when to apply them; step-by-step instructions for applying each refactoring; and an example illustrating how the refactoring works. The illustrative examples are written in Java, but the ideas are applicable to any object-oriented programming language.

Five Lines of Code - Christian Clausen - 2021-11-09
Five Lines of Code teaches refactoring that’s focused on concrete rules and getting any method down to five lines or less! There’s no jargon or tricky automated-testing-skills required, just easy guidelines and patterns illustrated by detailed code samples. In Five Lines of Code you will learn: The signs of bad code Improving code safely, even when you don’t understand it Balancing optimization and code quality The real-world benefits of this process Writing code that needs no comments Real-world practices for great refactoring Improving code quickly, even when you don’t understand it

The signs of bad code
Improving code safely, even when you don’t understand it
Balancing optimization and code quality
The real-world benefits of this process
Writing code that needs no comments
Real-world practices for great refactoring
Improving code quickly, even when you don’t understand it

Five Lines of Code teaches you clear and actionable refactoring rules that you can apply without relying on intuitive judgments such as “code anxiety.” Follow the expert perspectives of the author and code smells can be learned by following a concrete set of principles—you’ll learn when to refactor your code, what patterns to apply to what problem, and the code characteristics that indicate it’s time for a rework. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Every codebase includes mistakes and inefficiencies that you need to find and fix. Refactor the right way, and your code becomes elegant, easy to read, and easy to maintain. In this book, you’ll learn a unique approach to refactoring that implements all five lines or fewer. You’ll also discover a secret most senior devs know: sometimes it’s quicker to hammer out code and fix it later! About the book Five Lines of Code is a fresh look at refactoring for developers of all skill levels. In it, you’ll master author Christian Clausen’s innovative approach, learning concrete rules to get any method down to five lines—or less! You’ll learn when to refactor, specific refactoring patterns that apply to most common problems, and characteristics of code that should be deleted altogether. What’s inside-
The signs of bad code
Improving code safely, even when you don’t understand it
Balancing optimization and code quality
General Proprietary compiler practices About the reader For developers of all skill levels.

The Pragmatic Programmer - Andrew Hunt - 1999-10-20
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 Lakes, author of Large-Scale C++ Software Design “This is the book I would 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 for 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 Cleveland, 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 books.”—Ward Cunningham, Ward Cunningham Professional Practice

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 or less! There’s no jargon or tricky automated-testing-skills required, just easy guidelines and patterns illustrated by detailed code samples. In Five Lines of Code you will learn: The signs of bad code Improving code safely, even when you don’t understand it Balancing optimization and code quality

Principle-based-refactoring-learning-software-design-principles-by-applying-refactoring-rules
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.

The Pragmatic Programmer - Andrew Hunt - 1999-10-20

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 wish I would worry about writing because I would worry it being lost." —Kevin Buhr, 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 Programming "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." —Peter 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 me time and effort, 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, iRenaisance, Inc. "I would like to see this issued to every new employee at my company." —Chris Cleeand, 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.

Open Source for Knowledge and Learning Management: Strategies Beyond Tools - Lytras, Miltiadis D. - 2006-12-31

“This book presents learning and knowledge management from a point of view where the basic tools and applications are provided by open source technologies. It explains an intense orientation to the critical issues of the open source paradigm: open source tools, applications, social networks, and knowledge sharing in open source communities”–Provided by publisher.

Open Source for Knowledge and Learning Management: Strategies Beyond Tools - Lytras, Miltiadis D. - 2006-12-31

“This book presents learning and knowledge management from a point of view where the basic tools and applications are provided by open source technologies. It explains an intense orientation to the critical issues of the open source paradigm: open source tools, applications, social networks, and knowledge sharing in open source communities”–Provided by publisher.

International Conference on Innovative Computing and Communications - Ashish Khanna - 2021-08-17

This book includes high-quality research papers presented at the Fourth International Conference on Innovative Computing and Communication (ICICC 2021), which is held at the Shaheed Sukhdev College of Business Studies, University of Delhi, Delhi, India, on February 20–21, 2021. Introducing the innovative works of scientists, professors, research scholars, students and industrial experts in the field of computing and communication, the book promotes the transformation of fundamental research into institutional and industrialized research and the conversion of applied exploration into real-time applications.

International Conference on Innovative Computing and Communications - Ashish Khanna - 2021-08-17

This book includes high-quality research papers presented at the Fourth International Conference on Innovative Computing and Communication (ICICC 2021), which is held at the Shaheed Sukhdev College of Business Studies, University of Delhi, Delhi, India, on February 20–21, 2021. Introducing the innovative works of scientists, professors, research scholars, students and industrial experts in the field of computing and communication, the book promotes the transformation of fundamental research into institutional and industrialized research and the conversion of applied exploration into real-time applications.

Refactoring TypeScript - James Hickey - 2019-10-18

Discover various techniques to develop maintainable code and keep it in shape. Key Features Learn all about refactoring - why it is important and how to do it Discover easy ways to refactor code with examples Explore refactoring terminology Understand code that benefits from refactoring Use code that benefits from refactoring Introducing TypeScript A powerful and promising replacement for JavaScript Refactoring TypeScript explains how to spot bugs and remove them from your code. You'll start by seeing how wordy conditionals, methods, and null checks make code unhealthy and unstable. Whether it is identifying messy nested conditionals or removing unnecessary methods, this book will show various techniques to avoid these pitfalls and write code that is easier to understand, maintain, and test. By the end of the book, you’ll have learned some of the main causes of unhealthy code, tips to identify them and techniques to address them. What you will learn Spot and fix common code smells to create code that is easier to read and understand Discover ways to identify long methods and refactor them Create objects that keep your code flexible, maintainable, and testable Apply the Single Responsibility Principle to develop less-coupled code Discover how to combine different refactoring techniques Learn ways to solve the issues caused by everoning primitives Who this book is for This book is designed for programmers who are looking to explore various refactoring techniques to develop healthy and maintainable code. Some experience in JavaScript and TypeScript can help you easily grasp the concepts explained in this book.

Refactoring TypeScript - James Hickey - 2019-10-18

Discover various techniques to develop maintainable code and keep it in shape. Key Features Learn all about refactoring - why it is important and how to do it Discover easy ways to refactor code with examples Explore refactoring techniques that can be applied to most other programming languages Book Description Refactoring improves your code without changing its behavior. With refactoring, the best approach is to apply small targeted changes to a codebase. Instead of doing a huge sweeping change to your code, refactoring is better as a long-term and continuous enterprise. Refactoring TypeScript explains how to spot bugs and remove them from your code. You’ll start by seeing how wordy conditionals, methods, and null checks make code unhealthy and unstable. Whether it is identifying messy nested conditionals or removing unnecessary methods, this book will show various techniques to avoid these pitfalls and write code that is easier to understand, maintain, and test. By the end of the book, you’ll have learned some of the main causes of unhealthy code, tips to identify them and techniques to address them. What you will learn Spot and fix common code smells to create code that is easier to read and understand
Five Lines of Code teaches refactoring that’s focused on concrete rules and getting any method down to five lines maintainable, and testable. Apply the Single Responsibility Principle to develop less-coupled code. Discover how to combine different refactoring techniques. Learn ways to solve the issues caused by overusing primitives. This book is for those who are looking to explore various refactoring techniques to develop healthy and maintainable code. Some experience in JavaScript and TypeScript can help you easily grasp the concepts explained in this book.

Object-Oriented Design Knowledge: Principles, Heuristics and Best Practices - Garz s, Javier - 2006-07-31
“The software engineering community has advanced greatly in recent years and we currently have numerous defined items of knowledge, such as standards, methodologies, methods, metrics, techniques, languages, patterns, knowledge related to processes, concepts, etc. The main objective of this book is to give a unified and global vision about Micro-Architectural Design Knowledge, analyzing the main techniques, experiences and methods.”--Provided by publisher.

Object-Oriented Design Knowledge: Principles, Heuristics and Best Practices - Garz s, Javier - 2006-07-31
“The software engineering community has advanced greatly in recent years and we currently have numerous defined items of knowledge, such as standards, methodologies, methods, metrics, techniques, languages, patterns, knowledge related to processes, concepts, etc. The main objective of this book is to give a unified and global vision about Micro-Architectural Design Knowledge, analyzing the main techniques, experiences and methods.”--Provided by publisher.

Proceedings of the Sixth Collaborative Research Symposium on Security, E-learning, Internet and Networking - - 2010
Artificial General Intelligence - Jürgen Schmidhuber - 2011-07-19
This book constitutes the refereed proceedings of the 4th International Conference on Artificial General Intelligence, AGI 2011, held in Mountain View, CA, USA, in August 2011. The 28 revised full papers and 26 short papers were carefully reviewed and selected from 103 submissions. The papers are written by leading academic and industry researchers involved in scientific and engineering work and focus on the creation of AI systems possessing general intelligence at the human level and beyond.

Artificial General Intelligence - Jürgen Schmidhuber - 2011-07-19
This book constitutes the refereed proceedings of the 4th International Conference on Artificial General Intelligence, AGI 2011, held in Mountain View, CA, USA, in August 2011. The 28 revised full papers and 26 short papers were carefully reviewed and selected from 103 submissions. The papers are written by leading academic and industry researchers involved in scientific and engineering work and focus on the creation of AI systems possessing general intelligence at the human level and beyond.

Five Lines of Code - Christian Clausen - 2021-11-09
Five Lines of Code teaches refactoring that’s focused on concrete rules and getting any method down to five lines or less! There’s no jargon or tricky automated-testing skills required, just easy guidelines and patterns illustrated by detailed code samples. Improving existing code—refactoring—is one of the most common tasks you’ll face as a programmer. Written for working developers, Five Lines of Code teaches you clear and actionable refactoring rules that you can apply without relying on intuitive judgements. Thanks to this hands-on guide, you’ll find yourself programming faster while still delivering high-quality code that your teammates will love to work with. Five Lines of Code teaches refactoring that’s focused on concrete rules and getting any method down to five lines or less! There’s no jargon or tricky automated-testing skills required, just easy guidelines and patterns illustrated by detailed code samples. Chapter by chapter you’ll put techniques into action by refactoring a complete 2D puzzle game. Before you know it, you’ll be making serious and tangible improvements to your codebase. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications.

Lean IT - Principles to Practice - Akhilesh N Singh, Avinash Singh - 2018-06-12
Information Technology is one of the fastest evolving and widely spreading disciplines impacting our personal and professional lives across business, and social domains. Customers are looking for hassle-free, better, faster and cost effective IT solutions to solve their problems. Lean has emerged as an innovative business management system capable to deliver higher customer value. After grand success of lean in manufacturing and various service sectors, now Lean IT is being adopted by progressive IT organizations to enhance their global competitiveness and growth. Lean IT is the extension of lean manufacturing and lean service principles to the development and management of IT service. Its central concern, in the context of IT, is to create value for the customers and wealth for IT organizations through elimination of waste by improving processes, people, and work culture. This book is written by a team of two management consultants; one with hands-on expertise of Lean in manufacturing and service organizations and another with expertise in IT Management implementation. The purpose of this book is to trigger lean thinking in IT professionals.

Lean IT - Principles to Practice - Akhilesh N Singh, Avinash Singh - 2018-06-12
Information Technology is one of the fastest evolving and widely spreading disciplines impacting our personal and professional lives across business, and social domains. Customers are looking for hassle-free, better, faster and cost effective IT solutions to solve their problems. Lean has emerged as an innovative business management system capable to deliver higher customer value. After grand success of lean in manufacturing and various service sectors, now Lean IT is being adopted by progressive IT organizations to enhance their global competitiveness and growth. Lean IT is the extension of lean manufacturing and lean service principles to the development and management of IT services. Its central concern, in the context of IT, is to create value for the customers and wealth for IT organizations through elimination of waste by improving processes, people, and work culture. This book is written by a team of two management consultants; one with hands-on expertise of Lean in manufacturing and service organizations and another with expertise in IT Management implementation. The purpose of this book is to trigger lean thinking in IT professionals.

Agile and Lean Concepts for Teaching and Learning - David Parsons - 2018-10-24
This book explores the application of agile and lean techniques, originally from the field of software development and manufacturing, to various aspects of education. It covers a broad range of topics, including applying agile methods.
using team-based approaches to student-centred activities based on agile principles and processes. Demonstrating how agile and lean ideas can concretely be applied to education, the book offers practical guidance on how to apply these ideas in the classroom or lecture hall, as well as new concepts that could spark further research and development.

**Advances in Learning Software Organizations** - Alta.) Lso 200 (2004 Banff - 2004-06-14

Software-intensive organizations cannot help but learn. A software organization that does not learn will not exist for long, because the software market is continuously on the move, because of new customer demands and needs, and because of new competitor products and services. Software organizations must adapt quickly to this ever-changing environment, and the capability to adapt is one of the most important aspects of being. Smart organizations will attempt to predict future software demands, and develop a corresponding knowledge roadmap that identifies the capabilities needed over time in order to meet these demands. Organizational learning typically occurs when experienced organization members share their knowledge with colleagues, such that the organization as a whole can profit from the intellectual capital of its members. While knowledge is typically shared in an adhoc fashion by means of direct, face-to-face communication, a learning software organization will want to ensure that this knowledge sharing occurs in a systematic way, enabling it whenever and wherever it is needed. Since 1999, the annual International Workshop on Learning Software Organizations (LSO) has provided a forum that brings together academics and industry to discuss the advancements in and to address the questions of continuous learning in software-intensive organizations. Building upon existing work on knowledge management and organizational learning, the workshop series promotes interdisciplinary approaches from computer science and information systems, business, management and organization science as well as cognitive science.

**Agile Processes in Software Engineering and Extreme Programming** - Alberto Sillitti - 2011-05-02

This book contains the refereed proceedings of the 12th International Conference on Agile Software Development, XP 2011, held in Madrid, Spain, in May 2011. The year 2011 marked the 10th anniversary of the Agile Manifesto. In this spirit, the XP conference continued its fine tradition of promoting agility by disseminating new research and experiences. As introduced for XP 2010, there were again two different program committees, one for research papers and one for experience reports. Regarding the research papers, 11 out of 56 submissions were accepted as full papers; and as far as the experience reports were concerned, the respective number was 4 out of 17 submissions. In addition to these papers, this volume also includes the short research papers, the abstracts of the posters, the position papers of the PhD symposium, and the abstracts of the workshops.

**Thoughtful Machine Learning with Python** - Matthew Kirk - 2017-01-16

Gain the confidence you need to apply machine learning in your daily work. With this practical guide, author Matthew Kirk shows you how to integrate and test machine learning algorithms in your code, without the academic subtext. Featuring graphs and highlighted code examples throughout, the book features tests with Python’s Numpy, Pandas, Scikit-Learn, and SciPy data science libraries. If you’re a software engineer or business analyst interested in data science, this book will help you: Reference real-world examples to test each algorithm through engaging, hands-on exercises Apply test-driven development (TDD) to write and run tests before you start coding Explore techniques for improving your machine-learning models with data extraction and feature development Watch out for the risks of machine learning, such as underfitting or overfitting data. Work with K-Nearest Neighbors, neural networks, clustering, and other algorithms.