[PDF] Principle Based Refactoring Learning Software Design Principles By Applying Refactoring Rules

Right here, we have countless ebook principle based refactoring learning software design principles by applying refactoring rules and collections to check out. We additionally find the money for variant types and after that type of the books to browse. The agreeable book, fiction, history, novel, scientific research, as capably as various new sorts of books are readily reachable here.

As this principle based refactoring learning software design principles by applying refactoring rules, it ends in the works physical one of the favored books principle based refactoring learning software design principles by applying refactoring rules collections that we have. This is why you remain in the best website to see the incredible books to have.

**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 don’t! 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.

**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.

**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 Rewrite PART 3 BEYOND REFACTORING—IMPROVING PROJECT WORKFLOW AND INFRASTRUCTURE Automation and development environment Extending automation to test, staging, and production environments Modernizing the development, building, and deployment of legacy software Stop writing legacy 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
programming, and continuous integration Learn and apply the four publications. About the book Refactoring-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 decoupling 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 Automated infrastructure New tests for old code Modularizing monolithic projects About the reader This book is written for developers and team leaders 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 rewrite part 3 beyond refactoring—improving project workflowand infrastructure Automating the development environment Extending automation to test, staging, and production environments Modernizing the development, building, and deployment of legacy software Stop writing legacy code.

Refactoring - Martin Fowler - 1999

Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows exactly how to do it, step by step.

Refactoring - Martin Fowler - 1999

Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows exactly how to do it, step by step.


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. This book delves 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 the red, green, refactoring cycle of 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, Concurrency, SOLID principles, and code smells are all related Improved 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.


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. 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 the red, green, refactoring cycle of TDD and 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.


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. 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.
bibliographies with most chapters of the volumes are devoted to 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.


"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.

**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.

**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.

**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 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 were carefully reviewed and selected from 217 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 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

This book constitutes the refereed proceedings of the 9th Software Quality Days Conference, SWQD 2017, held in Vienna, Austria, in January 2017. The SWQD conference offers a range of comprehensive and valuable information by presenting new ideas from the latest research papers, keynote speeches by renowned academics and industry leaders, professional lectures, exhibits, and tutorials. The 4 full papers and 7 short papers presented in this volume were carefully reviewed and selected from 212 submissions. They were organized in topical sections named: model-driven development and configuration management; software development and quality assurance; software quality assurance in industry; crowdsourcing in software engineering; software testing and traceability; and process improvement. The book also contains one keynote talk in full paper length.

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

This book constitutes the refereed proceedings of the 9th Software Quality Days Conference, SWQD 2017, held in Vienna, Austria, in January 2017. The SWQD conference offers a range of comprehensive and valuable information by presenting new ideas from the latest research papers, keynote speeches by renowned academics and industry leaders, professional lectures, exhibits, and tutorials. The 4 full papers and 7 short papers presented in this volume were carefully reviewed and selected from 212 submissions. They were organized in topical sections named: model-driven development and configuration management; software development and quality assurance; software quality assurance in industry; crowdsourcing in software engineering; software testing and traceability; and process improvement. The book also contains one keynote talk in full paper length.


"This book provides the research and instruction used to develop and implement software quickly, in small iteration cycles, and in close cooperation with the customer in an adaptive way, making it possible to react to changes set by the constant changing business environment. It presents four values explaining extreme programming (XP), the most widely adopted agile methodology" - Provided by publisher.


"This book provides the research and instruction used to develop and implement software quickly, in small iteration cycles, and in close cooperation with the customer in an adaptive way, making it possible to react to changes set by the constant changing business environment. It presents four values explaining extreme programming (XP), the most widely adopted agile methodology" - Provided by publisher.

**Agile Software Engineering** - Orit Hazan - 2008-10-27

Overview and Goals The agile approach for software development has been applied more and more extensively since the mid nineties of the 20th century. Though there are only about ten years of accumulated experience using the agile approach, it is currently conceived as one of the mainstream approaches for software development. This book presents a complete software engineering course from the agile angle. Our intention is to 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: I The Human perspective,
seem to have any structure This book also includes a catalog of twenty-four interpersonal processes between teammates, customers, and management. I The Organizational perspective, which includes managerial and cultural aspects, and refers to software project management and control. I 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 to 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.

Agile Software Engineering - Ort Hazzan - 2008-10-27

Overview and Goals The agile approach for software development has been applied more and more extensively since the mid nineties of the 20th century. Though about ten years of accumulated experience with the agile approach, it is currently conceived as one of the mainstream approaches for software development. This book presents a complete software engineering course from the agile angle. Our intention is to 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: I The Human perspective, which includes cognitive and social aspects, and refers to learning and interpersonal processes between teammates, customers, and management. I The Organizational perspective, which includes managerial and cultural aspects, and refers to software project management and control. I 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 to 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 what makes code hard to 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—such as 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 what makes code hard to 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—such as 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 mis-application of principles or patterns. These smells tend to contribute heavily to technical debt – further time owed to fix projects thought to be complete – 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 diagnostic techniques for identifying the smells, illustrative examples of the problems that result Covers pragmatic techniques for refactoring design smells to manage technical debt and to create and maintain high-quality software in practice Presents insightful anecdotes and case studies drawn from the trenches of real-world projects.

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 mis-application of principles or patterns. These smells tend to contribute heavily to technical debt – further time owed to fix projects thought to be complete – 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 diagnostic techniques for identifying the smells, illustrative examples of the problems that result Covers pragmatic techniques for refactoring design smells to manage technical debt and to create and maintain high-quality software in practice Presents insightful anecdotes and case studies drawn from the trenches of real-world projects.

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 right combination of processes and technical excellence to address architectural challenges Explore a range of architectural techniques to modernize legacy systems Discover how to design and continuously improve well-architected sustainable software Using Book Description Many organizations have embraced Agile methodologies to transform their ability to rapidly respond to constantly changing customer demands. However, in this melee, many enterprises often neglect to invest in architects by presuming architecture is not an intrinsic element of Agile software development. Since the role of an architect is not pre-defined in Agile, many organizations struggle to position architects, often resulting in friction with other roles or a failure to
Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation - Torjeig Dingseay - 2014-12-06

This book constitutes the refereed proceedings of three international workshops held in Rome, Italy, in conjunction with the 15th International Conference on Agile Software Development, XP 2014, in May 2014. The workshops comprised Principles of Large-Scale Agile Development, Refactoring & Testing (RefTest 2014), and Estimations in the 21st Century Software Engineering (EstSE21 2014). The 13 revised full papers presented were carefully reviewed and selected from 28 submissions. In addition, an introduction and a keynote paper are included.

Leveraging Applications of Formal Methods, Verification and Validation: Engineering Principles - Tiziana Margaria - 2020-10-26
The three-volume set LNCS 12476 - 12478 constitutes the refereed proceedings of the 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, which was planned to take place during October 20-30, 2020, on Rhodes, Greece. The event itself was postponed to 2021 due to the COVID-19 pandemic. The papers presented were carefully reviewed and selected for inclusion in the proceedings. Each volume focuses on an individual topic with topical section headings within the volume: Part I, Verification Principles: Modularity and (De-)Composition in Verification; X-by-Construction: Correctness meets Probability; 30 Years of Statistical Model Checking; Verification and Validation of Concurrent and Distributed Systems. Part II, Engineering Principles: Automating Software Re-Engineering; Rigorous Engineering of Collective Adaptive Systems. Part III, Applications: Reliable Smart Contracts: State-of-the-art, Applications, Challenges and Future Directions; Automated Verification of Embedded Control Software; Formal methods for DIStributed Computing in future RAILway systems.

"This book provides fundamental research on the architecture of learning technology systems, discussing such issues as the common structures in LTS and solutions for specific forms such as knowledge-based, distributed, or adaptive applications of e-learning. Researchers, and scholars in the fields of learning content software development, computing and educational technologies, and e-learning will find it invaluable." --Provided by publisher.

Clean Code - Robert C. Martin - 2009
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.

Code Complete - Steve McConnell - 2004-06-09
 Widely considered one of the best practical guides to programming, Steve
some code out of a method to turn it into its own method, or even pushing 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

**Code Complete** - Steve McConnell - 2004-06-09

Wideley considered one of the best practical guides to programming, this book is filled with practical advice on how to design and write software that is both reliable and maintainable.

Intelligent Tutoring Systems - Mitsuori Ikeda - 2006-06-09

This book is the refereed proceedings of the 8th International Conference on Intelligent Tutoring Systems, held in Jhongli, Taiwan, June 2006. The book presents 67 revised full papers and 40 poster papers, together with abstracts of 6 keynote talks, organized in topical sections on assessment, authoring tools, baesian reasoning and decision-theoretic approaches, case-based and analogical reasoning, cognitive models, collaborative learning, e-learning and web-based intelligent tutoring systems, and more.

Intelligent Tutoring Systems - Mitsuori Ikeda - 2006-06-09

This book is the refereed proceedings of the 8th International Conference on Intelligent Tutoring Systems, ITS 2006, held in Jhongli, Taiwan, June 2006. The book presents 67 revised full papers and 40 poster papers, together with abstracts of 6 keynote talks, organized in topical sections on assessment, authoring tools, baesian reasoning and decision-theoretic approaches, case-based and analogical reasoning, cognitive models, collaborative learning, e-learning and web-based intelligent tutoring systems, and more.

Refactoring - Martin Fowler - 2012-03-09

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 "legacy," "non-optimal" applications. For several years, expert-level object programmers have employed a growing collection of techniques to improve the structural integrity and 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 whether 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 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

Refactoring - Martin Fowler - 2012-03-09

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 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 whether 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 generality Proper compiler practices 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 any method in five lines or fewer. You’ll also discover a secret most senior devns 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 generality Proper compiler practices About the reader For developers of all skill levels. Examples use easy-to-read Typescript, in the same style as Java and C#. About the author Christian Clausen works as a Technical Agile Coach, teaching teams how to refactor code. Table of Contents 1 Refactoring refactoring 2 Looking under the hood
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 defined code samples. In Five Lines of Code you will learn:

- The signs of bad code
- Improving code safety, even when you don’t understand it
- Balancing optimization and code generality
- Proper compiler practices
- About the reader

Table of Contents

1 Refactoring 2 Looping under the hood
2 Defend the data
3 Shatter long function
4 Make type codes work
5 Fuse similar code together
6 Love deleting code
7 Never be afraid to add code
8 Follow the structure
9 Stay away from comments
10 Make bad code look bad
11 Follow the structure
12 Avoid optimizations and generality
13 Make bad code look bad
14 Wrapping up

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

"I would like to see this issued to every new employee at my company." —Issac Clee, 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
understand, maintain, and test. By the end of the book, you’ll have learned

Beyond Tools - Lytras, Miliadis 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, Miliadis 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 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 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 overusing 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.

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

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: Beyond Tools - 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 downloading 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. 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 downloading 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. 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 downloading 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. 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 downloading faster while still delivering high-quality code that your teammates will love to work with.
Demonstrating how agile and lean can concretely be applied to programmer. Written for working developers, Five Lines of Code teaches you clear and active refactoring rules that you can apply without relying on intuitive judgments. 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.

Data Science - Pinnie Qin - 2020-08-20
This two volume set (CCIS 1257 and 1258) constitutes the refereed proceedings of the 6th International Conference of Pioneering Computer Scientists, Engineers and Educators, IPCSCEE 2020 held in Taiyuan, China, in September 2020. The 98 papers presented in these two volumes were carefully reviewed and selected from 392 submissions. The papers are organized in topical sections: database, machine learning, network, graphic images, system, natural language processing, security, algorithm, application, and education.

Data Science - Pinnie Qin - 2020-08-20
This two volume set (CCIS 1257 and 1258) constitutes the refereed proceedings of the 6th International Conference of Pioneering Computer Scientists, Engineers and Educators, IPCSCEE 2020 held in Taiyuan, China, in September 2020. The 98 papers presented in these two volumes were carefully reviewed and selected from 392 submissions. The papers are organized in topical sections: database, machine learning, network, graphic images, system, natural language processing, security, algorithm, application, and education.

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. New technologies have 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.

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 teaching and learning techniques in the classroom, incorporating lean thinking in educational workflows, and using team-based approaches to student-centred activities based on agile principles and processes.
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.

**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 results in a timely manner and by bringing together researchers and practitioners for a fruitful mutual exchange of 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

**Database and Expert Systems Applications** - Sven Hartmann - 2016-08-05
This two volume set LNCS 9827 and LNCS 9828 constitutes the refereed proceedings of the 27th International Conference on Database and Expert Systems Applications, DEXA 2016, held in Porto, Portugal, September 2016. The 39 revised full papers presented together with 29 short papers were carefully reviewed and selected from 137 submissions. The papers discuss a range of topics including: Temporal, Spatial, and High Dimensional Databases; Data Mining; Authenticity, Privacy, Security, and Trust; Data Clustering; Distributed and Big Data Processing; Decision Support Systems, and Learning; Data Streams; Data Integration, and Interoperability; Semantic Web, and Data Semantics; Social Networks, and Network Analysis; Linked Data; Data Analysis; NoSQL, NewSQL, Multimedia Data; Personal Information Management; Semantic Web and Ontologies; Database and Information System Architectures; Query Answering and Optimization; Information Retrieval, and Keyword Search; Data Modelling, and Uncertainty.

**Database and Expert Systems Applications** - Sven Hartmann - 2016-08-05
This two volume set LNCS 9827 and LNCS 9828 constitutes the refereed proceedings of the 27th International Conference on Database and Expert Systems Applications, DEXA 2016, held in Porto, Portugal, September 2016. The 39 revised full papers presented together with 29 short papers were carefully reviewed and selected from 137 submissions. The papers discuss a range of topics including: Temporal, Spatial, and High Dimensional Databases; Data Mining; Authenticity, Privacy, Security, and Trust; Data Clustering; Distributed and Big Data Processing; Decision Support Systems, and Learning; Data Streams; Data Integration, and Interoperability; Semantic Web, and Data Semantics; Social Networks, and Network Analysis; Linked Data; Data Analysis; NoSQL, NewSQL, Multimedia Data; Personal Information Management; Semantic Web and Ontologies; Database and Information System Architectures; Query Answering and Optimization; Information Retrieval, and Keyword Search; Data Modelling, and Uncertainty.