Extreme Programming Explained Embrace Change

Extreme Programming Installed explains the core principles of Extreme Programming and details each step in the XP development cycle. This book conveys the essence of the XP approach--techniques for implementation, obstacles likely to be encountered, and experience-based advice for successful execution.

Testing is a cornerstone of XP, as tests are written for every piece of code before it is programmed. This workbook helps testers learn XP, and XP devotees learn testing. This new book defines how an XP tester can optimally contribute to a project, including what testers should do, when they should do it, and how they should do it.

Extreme Programming has come a long way since its ?rst use in the C3 project almost 10 years ago. Agile methods have found their way into the mainstream, and at the end of last year we saw the second edition of Kent Beck’s book on Extreme Programming, containing a major refactoring of XP. This year, the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering took place June 18–23 in She?eld. As in the yearsbefore, XP 2005provideda unique forum for industry and academic professionals to discuss their needs and ideas on Extreme Programming and - ile methodologies. These proceedings re?ect the activities during the conference which ranged from presentation of research papers, invited talks, posters and demonstrations, panels and activity sessions, to tutorials and workshops. - cluded are also papers from the Ph.D. and Master’s Symposium which provided a forum for young researchers to present their results and to get feedback. Asvariedastheactivities werethe topicsofthe conferencewhichcoveredthe presentationofnewandimprovedpractices,empiricalstudies,experiencereports and case studies, and last but not least the social aspects of agile methods. The papers and the activities went through a rigorous reviewing process. Each paper was reviewed by at least three Program Committee members and wasdiscussedcarefullyamongtheProgramCommittee.Of62paperssubmitted, only 22 were accepted as full papers.

Extreme Programming ExplainedEmbrace ChangeAddison-Wesley Professional

"This remarkable book combines practical advice, ready-to-use techniques, anda deep understanding of why this is the right way to develop software. I haveseen software teams transformed by the ideas in this book." --Mike Cohn, author of Agile Estimating and Planning "As a lean practitioner myself, I have loved and used their first book for years.When this second book came out, I was delighted that it was even better. If youare interested in how lean principles can be useful for software developmentorganizations, this is the book you are looking for. The Poppendiecks offer abeautiful blend of history, theory, and practice." --Alan Shalloway, coauthor of Design Patterns Explained "I've enjoyed reading the book very much. I feel it might even be better than thefirst lean book by Tom and Mary, while that one was already exceptionallygood! Mary especially has a lot of knowledge related to lean techniques inproduct development and manufacturing. It's rare that these techniques areactually translated to software. This is something no other book does well(except their first book)." --Bas Vodde "The new book by Mary and Tom Poppendieck provides a well-written andcomprehensive introduction to lean principles and selected practices for softwaremanagers and engineers. It illustrates the application of the values andpractices with well-suited success stories. I enjoyed reading it." --Roman Pichler "In Implementing Lean Software Development, the Poppendiecks explore moredeeply the themes they introduced in Lean Software Development. They beginwith a compelling history of lean thinking, then move to key areas such asvalue, waste, and people. Each chapter includes exercises to help you apply keypoints. If you want a better understanding of how lean ideas can work withsoftware,

this book is for you." --Bill Wake, independent consultant In 2003, Mary and Tom Poppendieck's Lean Software Development introduced breakthrough development techniques that leverage Lean principles to deliver unprecedented agility and value. Now their widely anticipated sequel and companion guide shows exactly how to implement Lean software development, hands-on. This new book draws on the Poppendiecks' unparalleled experience helping development organizations optimize the entire software value stream. You'll discover the right questions to ask, the key issues to focus on, and techniques proven to work. The authors present case studies from leading-edge software organizations, and offer practical exercises for jumpstarting your own Lean initiatives. Managing to extend, nourish, and leverage agile practices Building true development teams, not just groups Driving quality through rapid feedback and detailed discipline Making decisions Just-in-Time, but no later Delivering fast: How PatientKeeper delivers 45 rock-solid releases per year Making tradeoffs that really satisfy customers Implementing Lean Software Development is indispensable to anyone who wants more effective development processes--managers, project leaders, senior developers, and architects in enterprise IT and software companies alike.

Provides information on eXtreme programming, or XP, a software development methodology.

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

A collection of current best practices and trends in reusable design patterns in software engineering, system design, and development, providing tested software design solutions for developers in all domains and organizations. Patterns are arranged by topic, with sections on general purpose design patterns and variations, and architectural, distribution, persistence, user-interface, programming, domain-specific, and process patterns, with a final chapter on a pattern language for pattern writing. Based on papers from American and European conferences held in 1996. Annotation copyrighted by Book News, Inc., Portland, OR

JUnit, created by Kent Beck and Erich Gamma, is an open source framework for test-driven development in any Java-based code. JUnit automates unit testing and reduces the effort required to frequently test code while developing it. While there are lots of bits of documentation all over the place, there isn't a go-to-manual that serves as a quick reference for JUnit. This Pocket Guide meets the need, bringing together all the bits of hard to remember information, syntax, and rules for working with JUnit, as well as delivering the insight and sage advice that can only come from a technology's creator. Any programmer who has written, or is writing, Java Code will find this book valuable. Specifically

it will appeal to programmers and developers of any level that use JUnit to do their unit testing in test-driven development under agile methodologies such as Extreme Programming (XP) [another Beck creation].

Corporate and commercial software-development teams all want solutions for one important problem—how to get their high-pressure development schedules under control. In RAPID DEVELOPMENT, author Steve McConnell addresses that concern head-on with overall strategies, specific best practices, and valuable tips that help shrink and control development schedules and keep projects moving. Inside, you'll find: A rapid-development strategy that can be applied to any project and the best practices to make that strategy work Candid discussions of great and not-so-great rapid-development practices—estimation, prototyping, forced overtime, motivation, teamwork, rapid-development languages, risk management, and many others A list of classic mistakes to avoid for rapid-development projects, including creeping requirements, shortchanged quality, and silver-bullet syndrome Case studies that vividly illustrate what can go wrong, what can go right, and how to tell which direction your project is going RAPID DEVELOPMENT is the real-world guide to more efficient applications development.

Apply what you know about extreme programming and object-oriented design to learning C# and the Microsoft® .NET Framework on the fly. Written by a leader in extreme programming, this book covers both high-level concepts and practical coding applications.

The first edition of "Extreme Programming Explained" is a classic. It won awards for its then-radical ideas for improving small-team development, such as having developers write automated tests for their own code and having the whole team plan weekly. Much has changed in five years. This completely rewritten second edition expands the scope of XP to teams of any size by suggesting a program of continuous improvement based on: five core values consistent with excellence in software development; eleven principles for putting those values into action; and, thirteen primary and eleven corollary practices to help you push development past its current business and technical limitations. Whether you have a small team that is already closely aligned with your customers or a large team in a gigantic or multinational organization, you will find in these pages a wealth of ideas to challenge, inspire, and encourage you and your team members to substantially improve your software development.

Ever since Extreme Programming burst on to the application development scene in 1998, it has been a lightning rod for controversy. In "Questioning Extreme Programming, " author McBreen puts this agile approach to application development under the microscope, and closely examines both sides of this heated debate.

This classic book is the definitive real-world style guide for better Smalltalk programming. This author presents a set of patterns that organize all the informal experience successful Smalltalk programmers have learned the hard way. When programmers understand these patterns, they can write much more effective code. The concept of Smalltalk patterns is introduced, and the book explains why they work. Next, the book introduces proven patterns for working with methods, messages, state, collections, classes and formatting. Finally, the book walks through a development example utilizing patterns. For programmers, project managers, teachers and students -- both new and experienced. This book presents a set of patterns that organize all the informal experience

of successful Smalltalk programmers. This book will help you understand these patterns, and empower you to write more effective code.

XPAgileUniverse2003isthethirdconferenceinaseriesrunninginNorthA- rica and attracting participants from all over the world who are interested in the research, development and application of agile software processes. Agile app- aches value people and interaction over processes and tools – moving software engineering from the process-oriented software development approaches of the 1990s towards people-oriented approaches that we are starting to see more and more in this decade. Agile approaches stress a holistic view of software deve- pers as being involved in analysis, design, implementation and testing activities, while more traditional, tayloristic approaches separate these tasks and assign them to di?erent “resources. ” Tayloristic approaches create knowledge-sharing problems as information gathered by one person needs to be handed over – usually in the form of documentation – to the next person in the chain. Agile approaches reduce the number of hand-o?s and, thus, decrease the amount of required documentation for knowledge sharing. While deemed a novelty only a few years ago, agile methods are now be- ming established in the software industry and are being applied in more and more application domains. While agile approaches move into the mainstream of software organizations, we are only now beginning to understand their bene?ts, areas of applicability, and also their dangers. This year’s conference will increase this understanding and provide a better base for industry practitioners as they assess the e?ectiveness of agile methods in their environment.

Accountability. Transparency. Responsibility. These are not words that are often applied to software development. In this completely revised introduction to Extreme Programming (XP), Kent Beck describes how to improve your software development by integrating these highly desirable concepts into your daily development process. The first edition of Extreme Programming Explained is a classic. It won awards for its then-radical ideas for improving small-team development, such as having developers write automated tests for their own code and having the whole team plan weekly. Much has changed in five years. This completely rewritten second edition expands the scope of XP to teams of any size by suggesting a program of continuous improvement based on: Five core values consistent with excellence in software development Eleven principles for putting those values into action Thirteen primary and eleven corollary practices to help you push development past its current business and technical limitations Whether you have a small team that is already closely aligned with your customers or a large team in a gigantic or multinational organization, you will find in these pages a wealth of ideas to challenge, inspire, and encourage you and your team members to substantially improve your software development. You will discover how to: Involve the whole team–XP style Increase technical collaboration through pair programming and continuous integration Reduce defects through developer testing Align business and technical decisions through weekly and quarterly planning Improve teamwork by setting up an informative, shared workspace You will also find many other concrete ideas for improvement, all based on a philosophy that emphasizes simultaneously increasing the humanity and effectiveness of software development. Every team can improve. Every team can begin improving today. Improvement is possible–beyond what we can currently imagine. Extreme Programming Explained, Second Edition, offers ideas to

fuel your improvement for years to come.

The 15th International Workshop on Conceptual Structures ICCS 2007 brings together numerous discussions between international groups of researchers from the field of Information and Communications Technology (ICT). At ICCS 2007 some of the world's best minds in information technology, arts, humanities and social science met to explore novel ways that ICT can augment human intelligence. The workshops include, Rough sets and data mining, and ubiquitous and collaborative computing.

Beck wants to encourage readers to re-examine their preconceptions of how software development ought to occur. He does just that in this overview of Extreme Programming, a controversial approach to software development which challenges the notion that the cost of changing a piece of software must rise dramatically over the course of time.

Write clean code that works with the help of this groundbreaking software method. Example-driven teaching is the basis of Beck's step-by-step instruction that will have readers using TDD to further their projects.

Algorithms play an important role in both the science and practice of computing. To optimally use algorithms, a deeper understanding of their logic and mathematics is essential. Beyond traditional computing, the ability to apply these algorithms to solve real-world problems is a necessary skill, and this is what this book focuses on.

An exploration of why people all over the world love to engage in pain on purpose--from dominatrices, religious ascetics, and ultramarathoners to ballerinas, icy ocean bathers, and sideshow performers Masochism is sexy, human, reviled, worshipped, and can be delightfully bizarre. Deliberate and consensual pain has been with us for millennia, encompassing everyone from Black Plague flagellants to ballerinas dancing on broken bones to competitive eaters choking down hot peppers while they cry. Masochism is a part of us. It lives inside workaholics, tattoo enthusiasts, and all manner of garden variety pain-seekers. At its core, masochism is about feeling bad, then better—a phenomenon that is long overdue for a heartfelt and hilarious investigation. And Leigh Cowart would know: they are not just a researcher and science writer—they're an inveterate, high-sensation seeking masochist. And they have a few questions: Why do people engage in masochism? What are the benefits and the costs? And what does masochism have to say about the human experience? By participating in many of these activities themselves, and through conversations with psychologists, fellow scientists, and people who seek pain for pleasure, Cowart unveils how our minds and bodies find meaning and relief in pain—a quirk in our programming that drives discipline and innovation even as it threatens to swallow us whole.

Stephens and Rosenberg examine XP in the context of existing methodologies and processes such as RUP, ICONIX, Spiral, RAD, DSDM, etc – and show how XP goals can be achieved using these existing processes.

We're losing tens of billions of dollars a year on broken software, and great new ideas such as agile development and Scrum don't always pay off. But there's hope. The nine software development practices in Beyond Legacy Code are designed to solve the problems facing our industry. Discover why these practices work, not just how they work, and dramatically increase the quality and maintainability of any software project. These nine practices could save the software industry. Beyond Legacy Code is filled with practical, hands-on advice and a common-sense exploration of why technical practices such as refactoring and test-first development are critical to building maintainable software. Discover how to avoid the pitfalls teams encounter when adopting these practices, and how to dramatically reduce the risk associated with building software--realizing significant savings in both the short and long term. With a deeper understanding of the principles behind the

practices, you'll build software that's easier and less costly to maintain and extend. By adopting these nine key technical practices, you'll learn to say what, why, and for whom before how; build in small batches; integrate continuously; collaborate; create CLEAN code; write the test first; specify behaviors with tests; implement the design last; and refactor legacy code. Software developers will find hands-on, pragmatic advice for writing higher quality, more maintainable, and bug-free code. Managers, customers, and product owners will gain deeper insight into vital processes. By moving beyond the old-fashioned procedural thinking of the Industrial Revolution, and working together to embrace standards and practices that will advance software development, we can turn the legacy code crisis into a true Information Revolution.

For those considering Extreme Programming, this book provides no-nonsense advice on agile planning, development, delivery, and management taken from the authors' many years of experience. While plenty of books address the what and why of agile development, very few offer the information users can apply directly.

Software Expert Kent Beck Presents a Catalog of Patterns Infinitely Useful for Everyday Programming Great code doesn’t just function: it clearly and consistently communicates your intentions, allowing other programmers to understand your code, rely on it, and modify it with confidence. But great code doesn’t just happen. It is the outcome of hundreds of small but critical decisions programmers make every single day. Now, legendary software innovator Kent Beck—known worldwide for creating Extreme Programming and pioneering software patterns and test-driven development—focuses on these critical decisions, unearthing powerful “implementation patterns” for writing programs that are simpler, clearer, better organized, and more cost effective. Beck collects 77 patterns for handling everyday programming tasks and writing more readable code. This new collection of patterns addresses many aspects of development, including class, state, behavior, method, collections, frameworks, and more. He uses diagrams, stories, examples, and essays to engage the reader as he illuminates the patterns. You’ll find proven solutions for handling everything from naming variables to checking exceptions.

Automated testing is a cornerstone of agile development. An effective testing strategy will deliver new functionality more aggressively, accelerate user feedback, and improve quality. However, for many developers, creating effective automated tests is a unique and unfamiliar challenge. xUnit Test Patterns is the definitive guide to writing automated tests using xUnit, the most popular unit testing framework in use today. Agile coach and test automation expert Gerard Meszaros describes 68 proven patterns for making tests easier to write, understand, and maintain. He then shows you how to make them more robust and repeatable--and far more cost-effective. Loaded with information, this book feels like three books in one. The first part is a detailed tutorial on test automation that covers everything from test strategy to in-depth test coding. The second part, a catalog of 18 frequently encountered "test smells," provides trouble-shooting guidelines to help you determine the root cause of problems and the most applicable patterns. The third part contains detailed descriptions of each pattern, including refactoring instructions illustrated by extensive code samples in multiple programming languages.

Build systems faster and more effectively with Mob Programming. Mob Programming is an approach to developing software that radically reduces defects and key-person dependencies by having a group of people work together at a single machine. See how to avoid the most common pitfalls that teams make when first starting out. Discover what it takes to create and support a successful mob. Now you can take collaborative programming to the next level with Mob Programming. Mob Programming is a natural extension of the popular Pair Programming concept, and is not restricted to a specific programming language or technology. It can be used by anyone who develops software, including dev leads, software developers, and agile coaches. The more people working on a bug or feature results in fewer dependencies on individuals, and overall increased learning for everyone involved. With more eyes on the code, you'll find you develop better

solutions with fewer defects. Set up your team for success by introducing Mob Programming in a way that benefits them. Create a good first Mobbing experience for your team with a template that avoids the common traps beginners may fall into. Master a collaborative and empathic mindset to help optimize the Mobbing experience. Learn how to make adjustments when things go wrong. Adapt your mobbing to different types of development tasks. Get management buy-in for your Mobbing experiment by demonstrating the benefits. Discover the equipment and resources you need, and how to adjust your workspace for an effective mob. Get important features to market sooner, squish bugs faster, and collaborate better today with Mob Programming. What You Need: All you need is three or more programmers, a meeting workspace that's large enough to accommodate your mob, and a computer on which to work.

Extreme Programming is the most exciting revolution to hit the software engineering industry in the last decade. But what exactly is XP? And how do you XP? Simply put, XP is about playing to win. If you are serious about becoming an agile organization, decreasing your time to market, keeping your development team happy, and improving the overall quality of your software, then XP is for you. Extreme Programming in Practice provides a candid, refreshing, insiders view of how an XP project works. The artifacts presented in this book are real, the user stories are real, and the anecdotes are real. The book represents all-access, uncensored XP. The authors have chosen example over explanation, so that you can personalize the tenets of XP and put them into practice on your next development project. The book is supported with sample code and test examples. You can learn how to emphasize planning in your project; deliver multiple iterations of your project (each with increasing business value); gather customer feedback as you build; and test the integrity of your code without halting your development efforts. The authors also provide a handy summary of more than a dozen lessons learned i

Ship It! is a collection of tips that show the tools andtechniques a successful project team has to use, and how to use themwell. You'll get quick, easy-to-follow advice on modernpractices: which to use, and when they should be applied. This bookavoids current fashion trends and marketing hype; instead, readersfind page after page of solid advice, all tried and tested in thereal world. Aimed at beginning to intermediate programmers, Ship It! will show you: Which tools help, and which don't How to keep a project moving Approaches to scheduling that work How to build developers as well as product What's normal on a project, and what's not How to manage managers, end-users and sponsors Danger signs and how to fix them Few of the ideas presented here are controversial or extreme; most experiencedprogrammers will agree that this stuff works. Yet 50 to 70 percent of allproject teams in the U.S. aren't able to use even these simple, well-acceptedpractices effectively. This book will help you get started. Ship It! begins by introducing the common technicalinfrastructure that every project needs to get the job done. Readerscan choose from a variety of recommended technologies according totheir skills and budgets. The next sections outline the necessarysteps to get software out the door reliably, using well-accepted,easy-to-adopt, best-of-breed practices that really work. Finally, and most importantly,

Ship It! presents commonproblems that teams face, then offers real-world advice on how tosolve them.
The co-author of Microsoft Secrets links issues related to strategy and organization to those of managing technology, arguing that companies must chose a business model that will capitalize on good times and survive more difficult periods, and presenting the success stories of such companies as IBM, Toshiba, and Motorola. 25,000 first printing.
This book constitutes the refereed proceedings of the 4th Conference on Extreme Programming and Agile Methods, XP/Agile Universe 2004, held in Calgary, Canada in August 2004. The 18 revised full papers presented together with summaries of workshops, panels, and tutorials were carefully reviewed and selected from 45 submissions. The papers are organized in topical sections on testing and integration, managing requirements and usability, pair programming, foundations of agility, process adaptation, and educational issues.
In 1994, Design Patterns changed the landscape of object-oriented development by introducing classic solutions to recurring design problems. In 1999, Refactoring revolutionized design by introducing an effective process for improving code. With the highly anticipated Refactoring to Patterns , Joshua Kerievsky has changed our approach to design by forever uniting patterns with the evolutionary process of refactoring. This book introduces the theory and practice of pattern-directed refactorings: sequences of low-level refactorings that allow designers to safely move designs to, towards, or away from pattern implementations. Using code from real-world projects, Kerievsky documents the thinking and steps underlying over two dozen pattern-based design transformations. Along the way he offers insights into pattern differences and how to implement patterns in the simplest possible ways. Coverage includes: A catalog of twenty-seven pattern-directed refactorings, featuring real-world code examples Descriptions of twelve design smells that indicate the need for this book’s refactorings General information and new insights about patterns and refactoring Detailed implementation mechanics: how low-level refactorings are combined to implement high-level patterns Multiple ways to implement the same pattern–and when to use each Practical ways to get started even if you have little experience with patterns or refactoring Refactoring to Patterns reflects three years of refinement and the insights of more than sixty software engineering thought leaders in the global patterns, refactoring, and agile development communities. Whether you’re focused on legacy or “greenfield” development, this book will make you a better software designer by helping you learn how to make important design changes safely and effectively.
You need to get value from your software project. You need it "free, now, and perfect." We can't get you there, but we can help you get to "cheaper, sooner, and better." This book leads you from the desire for value down to the specific activities that help good Agile projects deliver better software sooner, and at a lower cost. Using simple sketches and a few words, the author invites you to follow his path of learning and understanding from a half century of software

development and from his engagement with Agile methods from their very beginning. The book describes software development, starting from our natural desire to get something of value. Each topic is described with a picture and a few paragraphs. You're invited to think about each topic; to take it in. You'll think about how each step into the process leads to the next. You'll begin to see why Agile methods ask for what they do, and you'll learn why a shallow implementation of Agile can lead to only limited improvement. This is not a detailed map, nor a step-by-step set of instructions for building the perfect project. There is no map or instructions that will do that for you. You need to build your own project, making it a bit more perfect every day. To do that effectively, you need to build up an understanding of the whole process. This book points out the milestones on your journey of understanding the nature of software development done well. It takes you to a location, describes it briefly, and leaves you to explore and fill in your own understanding. What You Need: You'll need your Standard Issue Brain, a bit of curiosity, and a desire to build your own understanding rather than have someone else's detailed ideas poured into your head.

A guide to XP leads the developer, project manager, and team leader through the software development planning process, offering real world examples and tips for reacting to changing environments quickly and efficiently.

Extreme Programming (XP) is a significant departure from traditional software development methods, one that is ushering in a change for both developers and business people. It is an agile methodology, which enables highly productive teams to produce quality software from rapidly changing or unclear requirements. XP is disciplined software craftsmanship, elevating best practices in software analysis, design, testing, implementation, and project management to a new level. "Extreme Programming Applied" helps you begin using the principles behind this revolutionary concept. Even as the popularity of XP grows, many programmers and developers are still seeking practical advice on getting started. They find themselves in search of an XP roadmap, one that points to paths around the obstacles. "Extreme Programming Applied" is just that roadmap, a pragmatic guide to getting started with Extreme Programming. It helps programmers and project managers take their first steps toward applying the XP discipline. This book is not a tutorial, however. It uses real-world experience to educate readers about how to apply XP in their organizations. The authors offer guidelines for implementing XP, illustrating key points with valuable stories from successful XP pioneers. 0201616408B09172001

Learn what you need to succeed as a developer beyond the code. The lessons in this book will supercharge your career by sharing lessons and mistakes from real developers. Wouldn’t it be nice to learn from others’ career mistakes? “Soft” skills are crucial to success, but are haphazardly picked up on the job or, worse, never learned. Understanding these competencies and how to improve them will make you a more effective team member and a more attractive hire. This

book will teach you the key skills you need, including how to ask questions, how and when to use common tools, and how to interact with other team members. Each will be presented in context and from multiple perspectives so you'll be able to integrate them and apply them to your own career quickly. What You'll Learn Know when the best code is no code Understand what to do in the first month of your job See the surprising number of developers who can't program Avoid the pitfalls of working alone Who This Book Is For Anyone who is curious about software development as a career choice. You have zero to five years of software development experience and want to learn non-technical skills that can help your career. It is also suitable for teachers and mentors who want to provide guidance to their students and/or mentees.

Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress's highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words "at work" suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names. Having digested everyone's feedback, we selected 15 folks who've been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow Joe Armstrong: Inventor of Erlang Joshua Bloch: Author of the Java collections framework, now at Google Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger Douglas Crockford: JSON founder, JavaScript architect at Yahoo! L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1 Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal Dan Ingalls: Smalltalk implementor and designer Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler Donald Knuth: Author of The Art of Computer Programming and creator of TeX Peter Norvig: Director of Research at Google and author of the standard text on AI Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress Ken Thompson: Inventor of UNIX Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker

Written as instruction for pair programming newbies, with practical improvement tips for those experienced with the concept, this guide explores the operational aspects and unique fundamentals of pair programming; information such as furniture set-up, pair rotation, and weeding out bad pairs.

Agile is broken. Most Agile transformations struggle. According to an Allied Market Research study, "63% of respondents stated the failure of agile implementation in their organizations." The problems with Agile start at the top of most organizations with executive leadership not getting what agile is or even knowing the difference between success and failure in agile. Agile transformation is a journey, and most of that journey consists of people learning and trying new approaches in their own work. An agile organization can make use of coaches and training to improve their chances of success. But even then, failure remains because many Agile ideas are oversimplifications or interpreted in an extreme way, and many elements essential for success are missing. Coupled with other ideas that have been dogmatically forced on teams,

such as "agile team rooms", and "an overall inertia and resistance to change in the Agile community," the Agile movement is ripe for change since its birth twenty years ago. "Agile 2" represents the work of fifteen experienced Agile experts, distilled into Agile 2: The Next Iteration of Agile by seven members of the team. Agile 2 values these pairs of attributes when properly balanced: thoughtfulness and prescription; outcomes and outputs, individuals and teams; business and technical understanding; individual empowerment and good leadership; adaptability and planning. With a new set of Agile principles to take Agile forward over the next 20 years, Agile 2 is applicable beyond software and hardware to all parts of an agile organization including "Agile HR", "Agile Finance", and so on. Like the original "Agile", "Agile 2", is just a set of ideas - powerful ideas. To undertake any endeavor, a single set of ideas is not enough. But a single set of ideas can be a powerful guide.

Test-Driven Development (TDD) is now an established technique for delivering better software faster. TDD is based on a simple idea: Write tests for your code before you write the code itself. However, this "simple" idea takes skill and judgment to do well. Now there's a practical guide to TDD that takes you beyond the basic concepts. Drawing on a decade of experience building real-world systems, two TDD pioneers show how to let tests guide your development and “grow” software that is coherent, reliable, and maintainable. Steve Freeman and Nat Pryce describe the processes they use, the design principles they strive to achieve, and some of the tools that help them get the job done. Through an extended worked example, you’ll learn how TDD works at multiple levels, using tests to drive the features and the object-oriented structure of the code, and using Mock Objects to discover and then describe relationships between objects. Along the way, the book systematically addresses challenges that development teams encounter with TDD—from integrating TDD into your processes to testing your most difficult features. Coverage includes Implementing TDD effectively: getting started, and maintaining your momentum throughout the project Creating cleaner, more expressive, more sustainable code Using tests to stay relentlessly focused on sustaining quality Understanding how TDD, Mock Objects, and Object-Oriented Design come together in the context of a real software development project Using Mock Objects to guide object-oriented designs Succeeding where TDD is difficult: managing complex test data, and testing persistence and concurrency

Lean Software Development: An Agile Toolkit Adapting agile practices to your development organization Uncovering and eradicating waste throughout the software development lifecycle Practical techniques for every development manager, project manager, and technical leader Lean software development: applying agile principles to your organization In Lean Software Development, Mary and Tom Poppendieck identify seven fundamental "lean" principles, adapt them for the world of software development, and show how they can serve as the foundation for agile development approaches that work. Along the way, they introduce 22 "thinking tools" that can help you customize the right agile practices for any environment. Better, cheaper, faster software development. You can have all three–if you adopt the same lean principles that have already revolutionized manufacturing, logistics and product development. Iterating towards excellence: software development as an exercise in discovery Managing uncertainty: "decide as late as possible" by building change into the system. Compressing the value stream: rapid development, feedback, and improvement Empowering teams and individuals without compromising coordination Software with integrity: promoting coherence, usability, fitness, maintainability, and adaptability How to "see the whole"–even when your developers are scattered across multiple locations and contractors Simply put, Lean Software Development helps you refocus development on value, flow, and people–so you can achieve breakthrough quality, savings, speed, and business alignment.