

Adaptive Code Via Principles Developer

Using research in neurobiology, cognitive science and learning theory, this text loads patterns into your brain in a way that lets you put them to work immediately, makes you better at solving software design problems, and improves your ability to speak the language of patterns with others on your team.

By applying the principles in *Adaptive Code, Second Edition*, you can create code that adapts to new requirements and unforeseen scenarios without significant rework. Gary McLean Hall describes agile best practices, principles, and patterns for designing and writing code that can evolve more quickly and easily, with fewer errors, because it doesn't impede change. This concise, undogmatic book bridges theory and practice, demonstrating its principles and patterns with working C# code examples. Hall helps you: Organize and manage architectural dependencies Leverage best practice patterns -- and avoid anti-patterns Apply SOLID principles: single-responsibility, open/closed, Liskov substitution Manage interface versatility Perform unit testing and refactoring in tandem See how delegation and abstraction impact code adaptability Learn better ways to implement dependency interjection And much more Expanded and updated, this Second Edition adds new coverage of Kanban for BAU, Domain-Driven Design, Hexagonal Architecture, Test-Driven Development, and Test-First methodology. Hall also deepens and updates his discussions of unit testing, refactoring, and Pure Dependency Injection.

With the award-winning book *Agile Software Development: Principles, Patterns, and Practices*, Robert C. Martin helped bring Agile principles to tens of thousands of Java and C++ programmers. Now .NET programmers have a definitive guide to agile methods with this completely updated volume from Robert C. Martin and Micah Martin, *Agile Principles, Patterns, and Practices in C#*. This book presents a series of case studies illustrating the fundamentals of Agile development and Agile design, and moves quickly from UML models to real C# code. The introductory chapters lay out the basics of the agile movement, while the later chapters show proven techniques in action. The book includes many source code examples that are also available for download from the authors' Web site. Readers will come away from this book understanding Agile principles, and the fourteen practices of Extreme Programming Spiking, splitting, velocity, and planning iterations and releases Test-driven development, test-first design, and acceptance testing Refactoring with unit testing Pair programming Agile design and design smells The five types of UML diagrams and how to use them effectively Object-oriented package design and design patterns How to put all of it together for a real-world project Whether you are a C# programmer or a Visual Basic or Java programmer learning C#, a software development manager, or a business analyst, *Agile Principles, Patterns, and Practices in C#* is the first book you should read to understand agile software and how it applies to programming in the .NET Framework.

Often referred to as the "black art" because of its complexity and uncertainty, software estimation is not as difficult or puzzling as people think. In fact, generating accurate estimates is straightforward—once you understand the art of creating them. In his highly anticipated book, acclaimed author Steve McConnell unravels the mystery to successful software estimation—distilling academic information and real-world experience into a practical guide for working software professionals. Instead of arcane treatises and rigid modeling techniques, this guide highlights a proven set of procedures, understandable formulas, and heuristics that individuals and development teams can apply to their projects to help achieve estimation proficiency. Discover how to: Estimate schedule and cost—or estimate the functionality that can be delivered within a given time frame Avoid common software estimation mistakes Learn estimation techniques for you, your team, and your organization * Estimate specific project activities—including development, management, and defect correction Apply estimation approaches to any type of project—small or large, agile or traditional Navigate the shark-infested political waters that surround project estimates When many corporate software projects are failing, McConnell shows you what works for successful software estimation.

Adaptive Code via C# Agile coding with design patterns and SOLID principles Microsoft Press

In times of constant change, adaptive leadership is critical. This Harvard Business Review collection brings together the seminal ideas on how to adapt and thrive in challenging environments, from leading thinkers on the topic—most notably Ronald A. Heifetz of the Harvard Kennedy School and Cambridge Leadership Associates. The Heifetz Collection includes two classic books: *Leadership on the Line*, by Ron Heifetz and Marty Linsky, and *The Practice of Adaptive Leadership*, by Heifetz, Linsky, and Alexander Grashow. Also included is the popular Harvard Business Review article, "Leadership in a (Permanent) Crisis," written by all three authors. Available together for the first time, this collection includes full digital editions of each work. Adaptive leadership is a practical framework for dealing with today's mix of urgency, high stakes, and uncertainty. It has been used by individuals, organizations, businesses, and governments worldwide. In a world of challenging environments, adaptive leadership serves as a guide to distinguishing the essential from the expendable, beginning the meaningful process of adaption, and changing the status quo. Ronald A. Heifetz is a cofounder of the international leadership and consulting practice Cambridge Leadership Associates (CLA) and the founding director of the Center for Public Leadership at the Harvard Kennedy School. He is renowned worldwide for his innovative work on the practice and teaching of leadership. Marty Linsky is a cofounder of CLA and has taught at the Kennedy School for more than twenty-five years. Alexander Grashow is a Senior Advisor to CLA, having previously held the position of CEO. Microservices can have a positive impact on your enterprise—just ask Amazon and Netflix—but you can fall into many traps if you don't approach them in the right way. This practical guide covers the entire microservices landscape, including the principles, technologies, and methodologies of this unique, modular style of system building. You'll learn about the experiences of organizations around the globe that have successfully adopted microservices. In three parts, this book explains how these services work and what it means to build an application the Microservices Way. You'll explore a design-based approach to microservice architecture with guidance for implementing various elements. And you'll get a set of recipes and practices for meeting practical, organizational, and cultural challenges to microservice adoption. Learn how microservices can help you drive business objectives Examine the principles, practices, and culture that define microservice architectures Explore a model for creating complex systems and a design process for building a microservice architecture Learn the fundamental design concepts for individual microservices Delve into the operational elements of a microservices architecture, including containers and service discovery Discover how to handle the challenges of introducing microservice architecture in your organization Authored by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.

Five years and more than 100,000 copies after it was first published, it's hard to imagine anyone working in Web design who hasn't read Steve Krug's "instant classic" on Web usability, but people are still discovering it every day. In this second edition, Steve adds three new chapters in the same style as the original: wry and entertaining, yet loaded with insights and practical advice for novice and veteran alike. Don't be surprised if it completely changes the way you think about Web design. Three New Chapters! Usability as common courtesy -- Why people really leave Web sites Web Accessibility, CSS, and you -- Making sites usable and accessible Help! My boss wants me to _____. -- Surviving executive design whims "I thought usability was the enemy of design until I read the first edition of this book. Don't Make Me Think! showed me how to put myself in the position of the person who uses my site. After reading it over a couple of hours and putting its ideas to work for the past five years, I can say it has done more to improve my abilities as a Web designer than any other book. In this second edition, Steve Krug adds essential ammunition for those whose bosses, clients, stakeholders, and marketing managers insist on doing the wrong thing. If you design, write, program, own, or manage Web sites, you must read this book." -- Jeffrey Zeldman, author of *Designing with Web Standards*

Summary Dependency Injection Principles, Practices, and Patterns teaches you to use DI to reduce hard-coded dependencies between

application components. You'll start by learning what DI is and what types of applications will benefit from it. Then, you'll work through concrete scenarios using C# and the .NET framework to implement DI in your own projects. As you dive into the thoroughly-explained examples, you'll develop a foundation you can apply to any of the many DI libraries for .NET and .NET Core. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Dependency Injection (DI) is a great way to reduce tight coupling between software components. Instead of hard-coding dependencies, such as specifying a database driver, you make those connections through a third party. Central to application frameworks like ASP.NET Core, DI enables you to better manage changes and other complexity in your software. About the Book Dependency Injection Principles, Practices, and Patterns is a revised and expanded edition of the bestselling classic Dependency Injection in .NET. It teaches you DI from the ground up, featuring relevant examples, patterns, and anti-patterns for creating loosely coupled, well-structured applications. The well-annotated code and diagrams use C# examples to illustrate principles that work flawlessly with modern object-oriented languages and DI libraries. What's Inside Refactoring existing code into loosely coupled code DI techniques that work with statically typed OO languages Integration with common .NET frameworks Updated examples illustrating DI in .NET Core About the Reader For intermediate OO developers. About the Authors Mark Seemann is a programmer, software architect, and speaker who has been working with software since 1995, including six years with Microsoft. Steven van Deursen is a seasoned .NET developer and architect, and the author and maintainer of the Simple Injector DI library. Table of Contents PART 1 Putting Dependency Injection on the map The basics of Dependency Injection: What, why, and how Writing tightly coupled code Writing loosely coupled code PART 2 Catalog DI patterns DI anti-patterns Code smells PART 3 Pure DI Application composition Object lifetime Interception Aspect-Oriented Programming by design Tool-based Aspect-Oriented Programming PART 4 DI Containers DI Container introduction The Autofac DI Container The Simple Injector DI Container The Microsoft.Extensions.DependencyInjection DI Container

Agile has become today's dominant software development paradigm, but agile methods remain difficult to measure and improve. Essential Skills for the Agile Developer fills this gap from the bottom up, teaching proven techniques for assessing and optimizing both individual and team agile practices. Written by four principals of Net Objectives—one of the world's leading agile training and consulting firms—this book reflects their unsurpassed experience helping organizations transition to agile. It focuses on the specific actions and insights that can deliver the greatest design and programming improvements with economical investment. The authors reveal key factors associated with successful agile projects and offer practical ways to measure them. Through actual examples, they address principles, attitudes, habits, technical practices, and design considerations—and above all, show how to bring all these together to deliver higher-value software. Using the authors' techniques, managers and teams can optimize the whole organization and the whole product across its entire lifecycle. Essential Skills for the Agile Developer shows how to Perform programming by intention Separate use from construction Consider testability before writing code Avoid over- and under-design Succeed with Acceptance Test Driven Development (ATDD) Minimize complexity and rework Use encapsulation more effectively and systematically Know when and how to use inheritance Prepare for change more successfully Perform continuous integration more successfully Master powerful best practices for design and refactoring

Apply best practices for capturing, analyzing, and implementing software requirements through visual models—and deliver better results for your business. The authors—experts in eliciting and visualizing requirements—walk you through a simple but comprehensive language of visual models that has been used on hundreds of real-world, large-scale projects. Build your fluency with core concepts—and gain essential, scenario-based context and implementation advice—as you progress through each chapter. Transcend the limitations of text-based requirements data using visual models that more rigorously identify, capture, and validate requirements Get real-world guidance on best ways to use visual models—how and when, and ways to combine them for best project outcomes Practice the book's concepts as you work through chapters Change your focus from writing a good requirement to ensuring a complete system

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

Write code that can adapt to changes. By applying this book's principles, you can create code that accommodates new requirements and unforeseen scenarios without significant rewrites. Gary McLean Hall describes Agile best practices, principles, and patterns for designing and writing code that can evolve more quickly and easily, with fewer errors, because it doesn't impede change. Now revised, updated, and expanded, Adaptive Code, Second Edition adds indispensable practical insights on Kanban, dependency inversion, and creating reusable abstractions. Drawing on over a decade of Agile consulting and development experience, McLean Hall has updated his best-seller with deeper coverage of unit testing, refactoring, pure dependency injection, and more. Master powerful new ways to: • Write code that enables and complements Scrum, Kanban, or any other Agile framework • Develop code that can survive major changes in requirements • Plan for adaptability by using dependencies, layering, interfaces, and design patterns • Perform unit testing and refactoring in tandem, gaining more value from both • Use the "golden master" technique to make legacy code adaptive • Build SOLID code with single-responsibility, open/closed, and Liskov substitution principles • Create smaller interfaces to support more-diverse client and architectural needs • Leverage dependency injection best practices to improve code adaptability • Apply dependency inversion with the Stairway pattern, and avoid related anti-patterns About You This book is for programmers of all skill levels seeking more-practical insight into design patterns, SOLID principles, unit testing, refactoring, and related topics. Most readers will have programmed in C#, Java, C++, or similar object-oriented languages, and will be familiar with core procedural programming techniques.

In response to the US FDA's Critical Path Initiative, innovative adaptive designs are being used more and more in clinical trials due to their flexibility and efficiency, especially during early phase development. Handbook of Adaptive Designs in Pharmaceutical and Clinical Development provides a comprehensive and unified presentation of the princip

EDGE: The Agile Operating Model That Will Help You Successfully Execute Your Digital Transformation "[The authors'] passion for technology allows them to recognize that for most enterprises in the 21st century, technology is THE business. This is what really separates the EDGE approach. It is a comprehensive operating model with technology at its core." —From the Foreword by Heidi Musser, Executive Vice President and Principal Consultant, Leading Agile; retired, Vice President and CIO, USAA Maximum innovation happens at the edge of chaos: the messy, risky, and uncertain threshold between randomness and structure. Operating there is uncomfortable but it's where organizations "invent the future." EDGE is a set of fast, iterative, adaptive, lightweight, and value-driven tools to achieve digital transformation, and EDGE: Value-Driven Digital Transformation is your guide to using this operating model for innovation. Jim Highsmith is one of the world's leading agile pioneers and a coauthor of the Agile Manifesto. He, Linda Luu, and David Robinson know from their vast in-

the-trenches experience that sustainable digital transformation requires far more than adopting isolated agile practices or conventional portfolio management. This hard, indispensable work involves changing culture and mindset, and going beyond transforming the IT department. EDGE embraces an adaptive mindset in the face of market uncertainty, a visible, value-centered portfolio approach that encourages continual value linkages from vision to detailed initiatives, incremental funding that shifts as strategies evolve, collaborative decision-making, and better risk mitigation. This guide shows leaders how to use the breakthrough EDGE approach to go beyond incremental improvement in a world of exponential opportunities. Build an organization that adapts fast enough to thrive Clear away unnecessary governance processes, obsolete "command and control" leadership approaches, and slow budgeting/planning cycles Improve collaboration when major, fast-paced responses are necessary Continually optimize investment allocation and monitoring based on your vision and goals Register your product for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details. Write code that can adapt to changes. By applying this book's principles, you can create code that accommodates new requirements and unforeseen scenarios without significant rewrites. Gary McLean Hall describes Agile best practices, principles, and patterns for designing and writing code that can evolve more quickly and easily, with fewer errors, because it doesn't impede change. Now revised, updated, and expanded, Adaptive Code, Third Edition adds indispensable practical insights on Kanban, dependency inversion, and creating reusable abstractions. Drawing on over a decade of Agile consulting and development experience, McLean Hall has updated his best-seller with deeper coverage of unit testing, refactoring, pure dependency injection, and more. Master powerful new ways to: Write code that enables and complements Scrum, Kanban, or any other Agile framework Develop code that can survive major changes in requirements Plan for adaptability by using dependencies, layering, interfaces, and design patterns Perform unit testing and refactoring in tandem, gaining more value from both Use the "golden master" technique to make legacy code adaptive Build SOLID code with single-responsibility, open/closed, and Liskov substitution principles Create smaller interfaces to support more-diverse client and architectural needs Leverage dependency injection best practices to improve code adaptability Apply dependency inversion with the Stairway pattern, and avoid related anti-patterns About You This book is for programmers of all skill levels seeking more-practical insight into design patterns, SOLID principles, unit testing, refactoring, and related topics. Most readers will have programmed in C#, Java, C++, or similar object-oriented languages, and will be familiar with core procedural programming techniques.

This book explores the relationship between economic adaptation and long-run development, with particular emphasis on small, low-income economies. It also examines what makes for flexibility within an economy and how policy can affect an economy's ability to adapt to conditions over which it has no control. The premise is that all economies need to adapt to changing circumstances in order to achieve a reasonable pace of development. The author explains the forces to which economies need to respond, the attributes that increase an economy's capacity to adjust, the difficulties of adjustment, and what policy can do to facilitate adjustment. The author illustrates structure and flexibility within an economy and offers a guide to forming policy. Specific policy options are examined, among them using exchange rate fluctuations. The roles of government and markets in setting adjustment policies for industry, agriculture, and finance are explored. The study draws upon a wide range of material and avoids a narrowly economic point of view. The book is intended for use by economists working for or advising government agencies and for teachers and students of development economics. It includes an extensive reference list.

In OBJECT THINKING, esteemed object technologist David West contends that the mindset makes the programmer—not the tools and techniques. Delving into the history, philosophy, and even politics of object-oriented programming, West reveals how the best programmers rely on analysis and conceptualization—on thinking—rather than formal process and methods. Both provocative and pragmatic, this book gives form to what's primarily been an oral tradition among the field's revolutionary thinkers—and it illustrates specific object-behavior practices that you can adopt for true object design and superior results. Gain an in-depth understanding of: Prerequisites and principles of object thinking. Object knowledge implicit in eXtreme Programming (XP) and Agile software development. Object conceptualization and modeling. Metaphors, vocabulary, and design for object development. Learn viable techniques for: Decomposing complex domains in terms of objects. Identifying object relationships, interactions, and constraints. Relating object behavior to internal structure and implementation design. Incorporating object thinking into XP and Agile practice.

The rules and practices for Scrum—a simple process for managing complex projects—are few, straightforward, and easy to learn. But Scrum's simplicity itself—its lack of prescription—can be disarming, and new practitioners often find themselves reverting to old project management habits and tools and yielding lesser results. In this illuminating series of case studies, Scrum co-creator and evangelist Ken Schwaber identifies the real-world lessons—the successes and failures—culled from his years of experience coaching companies in agile project management. Through them, you'll understand how to use Scrum to solve complex problems and drive better results—delivering more valuable software faster. Gain the foundation in Scrum theory—and practice—you need to: Rein in even the most complex, unwieldy projects Effectively manage unknown or changing product requirements Simplify the chain of command with self-managing development teams Receive clearer specifications—and feedback—from customers Greatly reduce project planning time and required tools Build—and release—products in 30-day cycles so clients get deliverables earlier Avoid missteps by regularly inspecting, reporting on, and fine-tuning projects Support multiple teams working on a large-scale project from many geographic locations Maximize return on investment!

This volume focuses on two related questions that are central to both the psychology of mathematical thinking and learning and to the improvement of mathematics education: What is the nature of arithmetic expertise? How can instruction best promote it? Contributors from a variety of specialties, including cognitive, developmental, educational, and neurological psychology; mathematics education; and special education offer theoretical perspectives and much needed empirical evidence about these issues. As reported in this volume, both theory and research indicate that the nature of arithmetic expertise and how to best promote it are far more complex than conventional wisdom and many scholars, past and present, have suggested. The results of psychological, educational, and clinical studies using a wide range of arithmetic tasks and populations (including "normally" and atypically developing children, non-injured and brain-injured adults, and savants) all point to the same conclusion: The heart of arithmetic fluency, in general, and the flexible and creative use of strategies, in particular, is what is termed "adaptive expertise" (meaningful or conceptually based knowledge). The construction of adaptive expertise in mathematics is, for the first time, examined across various arithmetic topics and age groups. This book will be an invaluable resource for researchers and graduate students interested in mathematical cognition and learning (including mathematics educators, developmental and educational psychologists, and neuropsychologists), educators (including teachers, curriculum supervisors, and school administrators), and others interested in improving arithmetic instruction (including officials in national and local education departments, the media, and parents).

Agile coding with design patterns and SOLID principles As every developer knows, requirements are subject to change. But when you build adaptability into your code, you can respond to change more easily and avoid disruptive rework. Focusing on Agile programming, this book describes the best practices, principles, and patterns that enable you to create flexible, adaptive code—and deliver better business value. Expert guidance to bridge the gap between theory and practice Get grounded in Scrum: artifacts, roles, metrics, phases Organize and manage architectural dependencies Review best practices for patterns and anti-patterns Master SOLID principles: single-responsibility, open/closed, Liskov substitution Manage the versatility of interfaces for adaptive code Perform unit testing and refactoring in tandem See how delegation and abstraction impact code adaptability Learn best ways to implement dependency interjection Apply what you learn to a pragmatic, agile coding project Get code samples at: <http://github.com/garymclean/AdaptiveCode>

Get best-in-class engineering practices to help you write more-robust, bug-free code. Two Microsoft .NET development experts share real-

world examples and proven methods for optimizing the software development life cycle—from avoiding costly programming pitfalls to making your development team more efficient. Managed code developers at all levels will find design, prototyping, implementation, debugging, and testing tips to boost the quality of their code—today. Optimize each stage of the development process—from design to testing—and produce higher-quality applications. Use metaprogramming to reduce code complexity, while increasing flexibility and maintainability Treat performance as a feature—and manage it throughout the development life cycle Apply best practices for application scalability Employ preventative security measures to ward off malicious attacks Practice defensive programming to catch bugs before run time Incorporate automated builds, code analysis, and testing into the daily engineering process Implement better source-control management and check-in procedures Establish a quality-driven, milestone-based project rhythm—and improve your results!

It may surprise you to learn that Microsoft employs as many software testers as developers. Less surprising is the emphasis the company places on the testing discipline—and its role in managing quality across a diverse, 150+ product portfolio. This book—written by three of Microsoft's most prominent test professionals—shares the best practices, tools, and systems used by the company's 9,000-strong corps of testers. Learn how your colleagues at Microsoft design and manage testing, their approach to training and career development, and what challenges they see ahead. Most important, you'll get practical insights you can apply for better results in your organization. Discover how to: Design effective tests and run them throughout the product lifecycle Minimize cost and risk with functional tests, and know when to apply structural techniques Measure code complexity to identify bugs and potential maintenance issues Use models to generate test cases, surface unexpected application behavior, and manage risk Know when to employ automated tests, design them for long-term use, and plug into an automation infrastructure Review the hallmarks of great testers—and the tools they use to run tests, probe systems, and track progress efficiently Explore the challenges of testing services vs. shrink-wrapped software

Your process may be agile, but are you building agility directly into the code base? This book teaches .NET programmers how to give code the flexibility to adapt to changing requirements and customer demands by applying cutting-edge techniques, including SOLID principles, design patterns, and other industry best practices. Understand why composition is preferable to inheritance and how flexible the interface really can be Gain deep knowledge of key design patterns and anti-patterns, when to apply them, and how to give their code agility Bridge the gap between the theory behind SOLID principles, design patterns, and industry best practices by pragmatically solving real-world problems Get code samples written in upcoming version of Microsoft Visual C# Topics include: Agile with Scrum process; dependencies and layering; the interface; patterns and anti-patterns; introduction to SOLID principles, including open/closed and dependency interjection; and using application templates

Building Your I.T. Career A Complete Toolkit for a Dynamic Career in Any Economy Second Edition Break in. Move up. Earn more. Stay on top. Get the I.T. career edge you need right now! "They" say it's tougher now to build a great career in I.T. "They" complain about outsourcing, cutbacks, and the tough economy. Don't complain: act! Right this minute, outstanding I.T. jobs and careers are out there: You just have to know how to get them! This 100% I.T.-focused, up-to-the-minute toolkit delivers all the insider skills and insights you need to get your next great tech job now—and build lifelong success in the industry. It will help you plan your career, set achievable goals, organize them into practical action items, and make it happen! Totally updated for today's newest hiring trends, Building Your I.T. Career, Second Edition is packed with examples from real I.T. pros and hiring decision-makers, it will help you get in, get promoted, get raises, and stay in demand—one easy step at a time! -- Focus on the I.T. careers you'll be happiest and most successful in -- Discover what opportunity looks like today—and how to take advantage of it -- Adopt the proactive attitudes associated with I.T. career success -- Master the personal communication skills you need to get a job—and succeed when you have it -- Develop more effective cover letters and resumes, and interview brilliantly -- Break in to I.T. for the first time -- Build your social media and offline networks, and use them to supercharge your job search -- Negotiate salary and employment agreements that get you what you deserve -- Learn (and do) what it takes to get promoted -- Take advantage of telecommuting and consulting options -- Move into management (if that's what you want) -- Use mentors and career coaches effectively -- Become a high-priced hourly consultant -- Gain the personal financial discipline that liberates you to choose your best career options -- Make yourself nearly indispensable

Looks at a successful software project and provides details for software development for clients using object-oriented design and programming.

Traditional software development methods struggle to keep pace with the accelerated pace and rapid change of Internet-era development. Several "agile methodologies" have been developed in response -- and these approaches to software development are showing exceptional promise. In this book, Jim Highsmith covers them all -- showing what they have in common, where they differ, and how to choose and customize the best agile approach for your needs. KEY TOPICS: Highsmith begins by introducing the values and principles shared by virtually all agile software development methods. He presents detailed case studies from organizations that have used them, as well as interviews with each method's principal authors or leading practitioners. Next, he takes a closer look at the key features and techniques associated with each major Agile approach: Extreme Programming (XP), Crystal Methods, Scrum, Dynamic Systems Development Method (DSDM), Lean Development, Adaptive Software Development (ASD), and Feature-Driven Development (FDD). In Part III, Highsmith offers practical advice on customizing the optimal agile discipline for your own organization. MARKET: For all software developers, project managers, and other IT professionals seeking more flexible, effective approaches to developing software.

Building an elegant, functional website requires more than just knowing how to code. In Adaptive Web Design, Second Edition, you'll learn how to use progressive enhancement to build websites that work anywhere, won't break, are accessible by anyone—on any device—and are designed to work well into the future. This new edition of Adaptive Web Design frames even more of the web design process in the lens of progressive enhancement. You will learn how content strategy, UX, HTML, CSS, responsive web design, JavaScript, server-side programming, and performance optimization all come together in the service of users on whatever device they happen to use to access the web. Understanding progressive enhancement will make you a better web professional, whether you're a content strategist, information architect, UX designer, visual designer, front-end developer, back-end developer, or project manager. It will enable you to visualize experience as a continuum and craft interfaces that are capable of reaching more users while simultaneously costing less money to develop. When you've mastered the tenets and concepts of this book, you will see the web in a whole new way and gain web design superpowers that will make you invaluable to your employer, clients, and the web as a whole. Visit <http://adaptivewebdesign.info> to learn more.

Developmental evaluation (DE) offers a powerful approach to monitoring and supporting social innovations by working in partnership with program decision makers. In this book, eminent authority Michael Quinn Patton shows how to conduct evaluations within a DE framework. Patton draws on insights about complex dynamic systems, uncertainty, nonlinearity, and emergence. He illustrates how DE can be used for a range of purposes: ongoing program development, adapting effective principles of practice to local contexts, generating innovations and taking them to scale, and facilitating rapid response in crisis situations. Students and practicing evaluators will appreciate the book's extensive case examples and stories, cartoons, clear writing style, "closer look"

sidebars, and summary tables. Provided is essential guidance for making evaluations useful, practical, and credible in support of social change.

Covers topics such as the importance of secure systems, threat modeling, canonical representation issues, solving database input, denial-of-service attacks, and security code reviews and checklists.

You think you have a winning strategy. But do you? Executives are bombarded with bestselling ideas and best practices for achieving competitive advantage, but many of these ideas and practices contradict each other. Should you aim to be big or fast? Should you create a blue ocean, be adaptive, play to win—or forget about a sustainable competitive advantage altogether? In a business environment that is changing faster and becoming more uncertain and complex almost by the day, it's never been more important—or more difficult—to choose the right approach to strategy. In this book, The Boston Consulting Group's Martin Reeves, Knut Haanæs, and Janmejaya Sinha offer a proven method to determine the strategy approach that is best for your company. They start by helping you assess your business environment—how unpredictable it is, how much power you have to change it, and how harsh it is—a critical component of getting strategy right. They show how existing strategy approaches sort into five categories—Be Big, Be Fast, Be First, Be the Orchestrator, or simply Be Viable—depending on the extent of predictability, malleability, and harshness. In-depth explanations of each of these approaches will provide critical insight to help you match your approach to strategy to your environment, determine when and how to execute each one, and avoid a potentially fatal mismatch. Addressing your most pressing strategic challenges, you'll be able to answer questions such as: • What replaces planning when the annual cycle is obsolete? • When can we—and when should we—shape the game to our advantage? • How do we simultaneously implement different strategic approaches for different business units? • How do we manage the inherent contradictions in formulating and executing different strategies across multiple businesses and geographies? Until now, no book brings it all together and offers a practical tool for understanding which strategic approach to apply. Get started today.

This how-to resource provides leaders with a concrete framework for a strategic improvement plan, helping educators link the "principles" to "processes" of planning. Packed with key takeaways and additional resources, this book provides the concrete tools to design a strong strategy for improvement and enables educational leaders to think constructively about why we plan, what an effective strategic plan should contain, and how to create meaningful dialogue to support plan development, implementation, and monitoring for continuous improvement. The Strategy Playbook for Educational Leaders provides superintendents, central office staff, principals, and teacher leaders with the opportunity to reframe the process of their strategic planning and breathe new life into the activity.

Partial Contents1: Software Ascents- Components of Adaptive Software Development2: Thriving at the Edge of Chaos- The Adaptive Development Model3: The Project Mission- Identify the Mission- Create Mission Artifacts- Share Mission Values- Focus on Results4: Planning Adaptive Development Cycles- Adaptive Planning Techniques- The Evolving World of Components5: Great Groups and the Ability to Collaborate- Using Complexity Concepts to Improve Collaboration- Joint Application Development6: Learning: Models, Techniques, and Cycle Review Practices- Software Inspections- Project Postmortems7: Why Even Good Managers Cause Projects to Fail- Disruptive Technologies- No Silver Bullet8: Adaptive Management- The Progression from Process to Pattern9: Workstate Life Cycle Management- Managing Workflow in an Adaptive Environment10: Structural Collaboration- Eight Guidelines for Applying Rigor to Project Work11: Managing Project Time Cycles- Plan the Project12: Dawdling, McLuhan, and Thin Air- Organizational Growth- Surviving in Thin AirBibliographyIndex

This book teaches you all the essential knowledge required to learn and apply time-proven SOLID principles of object-oriented design and important design patterns in ASP.NET Core 1.0 (formerly ASP.NET 5) applications. You will learn to write server-side as well as client-side code that makes use of proven practices and patterns. SOLID is an acronym popularized by Robert Martin used to describe five basic principles of good object-oriented design--Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation and Dependency Inversion. This book covers all five principles and illustrates how they can be used in ASP.NET Core 1.0 applications. Design Patterns are time proven solutions to commonly occurring software design problems. The most well-known catalog of design patterns comes from Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, the so-called as GoF patterns (Gang of Four patterns). This book contains detailed descriptions of how to apply Creational, Structural and Behavioral GoF design patterns along with some Patterns of Enterprise Application Architecture. Popular JavaScript patterns are covered, along with working examples of all these patterns in ASP.NET Core 1.0 and C# are included. What You Will Learn: How to apply SOLID principles to ASP.NET applications How to use Gang of Four (GoF) design patterns in ASP.NET applications Techniques for applying Patterns of Enterprise Application Architecture cataloged by Martin Fowler in ASP.NET applications How to organize code and apply design patterns in JavaScript Who This Book Is For:This book is for ASP.NET developers familiar with ASP.NET Core 1.0, C# and Visual Studio.

Have you ever felt frustrated working with someone else's code? Difficult-to-maintain source code is a big problem in software development today, leading to costly delays and defects. Be part of the solution. With this practical book, you'll learn 10 easy-to-follow guidelines for delivering C# software that's easy to maintain and adapt. These guidelines have been derived from analyzing hundreds of real-world systems. Written by consultants from the Software Improvement Group (SIG), this book provides clear and concise explanations, with advice for turning the guidelines into practice. Examples for this edition are written in C#, while our companion Java book provides clear examples in that language. Write short units of code: limit the length of methods and constructors Write simple units of code: limit the number of branch points per method Write code once, rather than risk copying buggy code Keep unit interfaces small by extracting parameters into objects Separate concerns to avoid building large classes Couple architecture components loosely Balance the number and size of top-level components in your code Keep your codebase as small as possible Automate tests for your codebase Write clean code, avoiding "code smells" that indicate deeper problems Streamline project workflow with expert agile implementation The Project Management Profession is beginning to go through rapid and profound transformation due to the widespread adoption of agile methodologies. Those changes are likely to dramatically change the role of project managers in many environments as we have known them and raise the bar for the entire project management profession; however, we are in the early stages of that transformation and there is a lot of confusion about the impact it has on project managers: There are many stereotypes and misconceptions that exist about both Agile and traditional plan-driven project management, Agile and traditional project management principles and practices are treated as separate and independent domains of knowledge with little or no integration between the two

andsometimes seen as in conflict with each other Agile and "Waterfall" are thought of as two binary, mutually-exclusive choices and companies sometimes try to force-fit their business and projects to one of those extremes when the right solution is to fit the approach to the project It's no wonder that many Project Managers might be confused by all of this! This book will help project managers unravel a lot of the confusion that exists; develop a totally new perspective to see Agile and traditional plan-driven project management principles and practices in a new light as complementary to each other rather than competitive; and learn to develop an adaptive approach to blend those principles and practices together in the right proportions to fit any situation. There are many books on Agile and many books on traditional project management but what's very unique about this book is that it takes an objective approach to help you understand the strengths and weaknesses of both of those areas to see how they can work synergistically to improve project outcomes in any project. The book includes discussion topics, real world case studies, and sample enterprise-level agile frameworks that facilitate hands-on learning as well as an in-depth discussion of the principles behind both Agile and traditional plan-driven project management practices to provide a more thorough level of understanding.

Thoroughly rewritten for today's web environment, this bestselling book offers a fresh look at a fundamental topic of web site development: navigation design. Amid all the changes to the Web in the past decade, and all the hype about Web 2.0 and various "rich" interactive technologies, the basic problems of creating a good web navigation system remain. Designing Web Navigation demonstrates that good navigation is not about technology—it's about the ways people find information, and how you guide them. Ideal for beginning to intermediate web designers, managers, other non-designers, and web development pros looking for another perspective, Designing Web Navigation offers basic design principles, development techniques and practical advice, with real-world examples and essential concepts seamlessly folded in. How does your web site serve your business objectives? How does it meet a user's needs? You'll learn that navigation design touches most other aspects of web site development. This book: Provides the foundations of web navigation and offers a framework for navigation design Paints a broad picture of web navigation and basic human information behavior Demonstrates how navigation reflects brand and affects site credibility Helps you understand the problem you're trying to solve before you set out to design Thoroughly reviews the mechanisms and different types of navigation Explores "information scent" and "information shape" Explains "persuasive" architecture and other design concepts Covers special contexts, such as navigation design for web applications Includes an entire chapter on tagging While Designing Web Navigation focuses on creating navigation systems for large, information-rich sites serving a business purpose, the principles and techniques in the book also apply to small sites. Well researched and cited, this book serves as an excellent reference on the topic, as well as a superb teaching guide. Each chapter ends with suggested reading and a set of questions that offer exercises for experiencing the concepts in action.

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.

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.

A software architect's digest of core practices, pragmatically applied Designing effective architecture is your best strategy for managing project complexity—and improving your results. But the principles and practices of software architecting—what the authors call the "science of hard decisions"—have been evolving for cloud, mobile, and other shifts. Now fully revised and updated, this book shares the knowledge and real-world perspectives that enable you to design for success—and deliver more successful solutions. In this fully updated Second Edition, you will: Learn how only a deep understanding of domain can lead to appropriate architecture Examine domain-driven design in both theory and implementation Shift your approach to code first, model later—including multilayer architecture Capture the benefits of prioritizing software maintainability See how readability, testability, and extensibility lead to code quality Take a user experience (UX) first approach, rather than designing for data Review patterns for organizing business logic Use event sourcing and CQRS together to model complex business domains more effectively Delve inside the persistence layer,

including patterns and implementation.

[Copyright: f80cc7ffe310ad68125c65f94b0847ee](#)