Access Free The Art Of Software Support Design And Operation Of Suport Centers And Help Desks Free Download Pdf

The Art of Software Testing Software Estimation The Art of Software Security Assessment The Art of Software Modeling The Art of Software Security Testing The Art and Science of Analyzing Software Data Software Conflict 2.0 The Art of Software Architecture The Art of Agile Development The Art of R Programming The Software Arts Art of Software Design Effective UI The Art of Differentiating Computer Programs The Art of Readable Code The Art of Software Innovation The Art of Software Testing UML in Practice The Art of R Programming The Art of Software Testing The Art of UNIX Programming The Art of Support Testing Computer Software The Art of Software Testing, Second Edition THE ART OF SOFTWARE TESTING, 2ND ED The Art of Business Value The Art of Software Thermal Management for Embedded Systems Software State-of-the-art The Art of Software Testing The Art of Type and Typography Interface Design Embedded Systems: World Class Designs Programming Cultures Software Testing The Art Of Software Architecture Design Methods-Misl-Wiley Series The Art of Lean Software Development Making Things Happen The Art of Software Support The Art of Analysis Readme

R is the world's most popular language for developing statistical software: Archaeologists use it to track the spread of ancient civilizations, drug companies use it to discover which medications are safe and effective, and actuaries use it to assess financial risks and keep economies running smoothly. The Art of R Programming takes you on a guided tour of software development with R, from basic types and data structures to advanced topics like closures, recursion, and anonymous functions. No statistical knowledge is required, and your programming skills can range from hobbyist to pro. Along the way, you'll learn about functional and object-oriented programming, running mathematical simulations, and rearranging complex data into simpler, more useful formats. You'll also learn to: –Create artful graphs to visualize complex data sets and functions –Write more efficient code using parallel R and vectorization –Interface R with C/C++ and Python for increased speed or functionality –Find new R packages for text analysis, image manipulation, and more –Squash annoying bugs with advanced debugging techniques Whether you're designing aircraft, forecasting the weather, or you just need to tame your data, The Art of R Programming is your guide to harnessing the power of statistical computing. This long-awaited revision of a bestseller provides a practical discussion of the nature and aims of software testing. You'll find the latest methodologies for the design of effective test cases, including information on psychological and economic principles, managerial aspects, test tools, high-order testing, code inspections, and debugging. Accessible, comprehensive, and always practical, this edition provides the key information you need to test successfully, whether a novice or a working programmer. Buy your copy today and end up with fewer bugs tomorrow. This is the first entry-level book on algorithmic (also known as automatic) differentiation (AD), providing fundamental rules for the generation of first- and higher-order tangent-linear and adjoint code. The author covers the mathematical underpinnings as well as how to apply these observations to real-world numerical simulation programs. Readers will find: examples and exercises, including hints to solutions; the prototype AD tools dco and dcc for use with the examples and exercises; first- and higher-order tangent-linear and adjoint modes for a limited subset of C/C++, provided by the derivative code compiler dcc; a supplementary website containing sources of all software discussed in the book, additional exercises and comments on their solutions (growing over the coming years), links to other sites on AD, and errata. Famed author Jack Ganssle has selected the very best embedded systems design material from the Newnes portfolio. The result is a book covering the gamut of embedded design, from hardware to software to integrated embedded systems, with a strong pragmatic emphasis. The papers were selected from more than a dozen sources, including IEEE Computer, Software -- Practice & Experience, IEEE Transactions on Software Engineering, and Communications of the ACM. An alternative history of software that places the liberal arts at the very center of software's evolution. In The Software Arts, Warren Sack offers an alternative history of computing that places the arts at the very center of software's evolution. Tracing the origins of software to eighteenth-century French encyclopedists' step-by-step descriptions of how things were made in the workshops of artists and artisans, Sack shows that programming languages are the offspring of an effort to describe the mechanical arts in the language of the liberal arts. Sack offers a reading of the texts of computing—code, algorithms, and technical papers—that emphasizes continuity between prose and programs. He translates concepts and categories from the liberal and mechanical arts—including logic, rhetoric, grammar, learning, algorithm, language, and simulation—into terms of computer science and then considers their further translation into popular culture, where they circulate as forms of digital life. He considers, among other topics, the “arithmetization” of knowledge that presaged digitization; today's multitude of logics; the history of demonstration, from deduction to newer forms of persuasion; and the post-Chomsky absence of meaning in grammar. With The Software Arts, Sack invites artists and humanists to see how their ideas are at the root of software and invites computer scientists to envision themselves as artists and humanists. "Do you really understand what business value is? Information technology can and should deliver business value. But the Agile literature has paid scant attention to what business value means—and how to know whether or not you are delivering it. This problem becomes ever more critical as you push value delivery toward autonomous teams and away from requirements “tossed over the wall” by business stakeholders. An empowered team needs to understand its goal! Playful and thought-provoking, The Art of Business Value explores what business value means, why it matters, and how it should affect your software development and delivery practices. More than any other IT delivery approach, DevOps (and Agile thinking in general) makes business value a central concern. This book examines the role of business value in software and makes a compelling case for why a clear understanding of business value will change the way you deliver software. This book will make you think deeply about not only what it means to deliver value but also the relationship of the IT organization to the rest of the enterprise. It will give you the language to discuss value with the business, methods to cut through bureaucracy and strategies for incorporating Agile teams and culture into the enterprise. Most of all, this book will startle you into new ways of thinking about the cutting-edge of Agile practice and where it may lead." Modeling complex systems is a difficult challenge and all too often one in which modelers are left to their own devices. Using a multidisciplinary approach, The Art of Software Modeling covers theory, practice, and presentation in detail. It focuses on the importance of model creation and demonstrates how to create meaningful models. Presenting three self-contained sections, the text examines the background of modeling and frameworks for organizing information. It identifies techniques for researching and capturing client and system information and addresses the challenges of presenting models to specific audiences. Using concepts from art theory and aesthetics, this broad-based approach encompasses software practices, cognitive science, and information presentation. The book also looks at perception and cognition of diagrams, view composition, color theory, and presentation techniques. Providing practical methods for investigating and organizing complex information, The Art of Software Modeling demonstrates the effective use of modeling techniques to improve the development process and establish a functional, useful, and maintainable software system. 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. R is the world's most popular language for developing statistical software: Archaeologists use it to track the spread of ancient civilizations, drug companies use it to discover which medications are safe and effective, and actuaries use it to assess financial risks and keep economies running smoothly. The Art of R Programming takes you

on a guided tour of software development with R, from basic types and data structures to advanced topics like closures, recursion, and anonymous functions. No statistical knowledge is required, and your programming skills can range from hobbyist to pro. Along the way, you'll learn about functional and object-oriented programming, running mathematical simulations, and rearranging complex data into simpler, more useful formats. You'll also learn to: –Create artful graphs to visualize complex data sets and functions –Write more efficient code using parallel R and vectorization –Interface R with C/C++ and Python for increased speed or functionality –Find new R packages for text analysis, image manipulation, and more –Squash annoying bugs with advanced debugging techniques Whether you're designing aircraft, forecasting the weather, or you just need to tame your data, The Art of R Programming is your guide to harnessing the power of statistical computing. This succinct book explains how you can apply the practices of Lean software development to dramatically increase productivity and quality. Based on techniques that revolutionized Japanese manufacturing, Lean principles are being applied successfully to product design, engineering, the supply chain, and now software development. With The Art of Lean Software Development, you'll learn how to adopt Lean practices one at a time rather than taking on the entire methodology at once. As you master each practice, you'll see significant, measurable results. With this book, you will: Understand Lean's origins from Japanese industries and how it applies to software development Learn the Lean software development principles and the five most important practices in detail Distinguish between the Lean and Agile methodologies and understand their similarities and differences Determine which Lean principles you should adopt first, and how you can gradually incorporate more of the methodology into your process Review hands-on practices, including descriptions, benefits, trade-offs, and roadblocks Learn how to sell these principles to management The Art of Lean Software Development is ideal for busy people who want to improve the development process but can't afford the disruption of a sudden and complete transformation. The Lean approach has been yielding dramatic results for decades, and with this book, you can make incremental changes that will produce immediate benefits. "This book presents Lean practices in a clear and concise manner so readers are motivated to make their software more reliable and less costly to maintain. I recommend it to anyone looking for an easy-to-follow guide to transform how the developer views the process of writing good software."-- Bryan Wells, Boeing Intelligence & Security Sytems Mission System "If you're new to Lean software development and you're not quite sure where to start, this book will help get your development process going in the right direction, one step at a time."-- John McClenning, software development lead, Aclara This updated and reorganized fourth edition of Software Testing: A Craftsman's Approach applies the strong mathematics content of previous editions to a coherent treatment of Model-Based Testing for both code-based (structural) and specification-based (functional) testing. These techniques are extended from the usual unit testing discussions to full coverage of less understood levels integration and system testing. The Fourth Edition: Emphasizes technical inspections and is supplemented by an appendix with a full package of documents required for a sample Use Case technical inspection Introduces an innovative approach that merges the Event-Driven Petri Nets from the earlier editions with the "Swim Lane" concept from the Unified Modeling Language (UML) that permits model-based testing for four levels of interaction among constituents in a System of Systems Introduces model-based development and provides an explanation of how to conduct testing within model-based development environments Presents a new section on methods for testing software in an Agile programming environment Explores test-driven development, reexamines all-pairs testing, and explains the four contexts of software testing Thoroughly revised and updated, Software Testing: A Craftsman's Approach, Fourth Edition is sure to become a standard reference for those who need to stay up to date with evolving technologies in software testing. Carrying on the tradition of previous editions, it will continue to serve as a valuable reference for software testers, developers, and engineers. People expect effortless, engaging interaction with desktop and web applications, but producing software that generates enjoyable user experiences is much harder than many companies anticipate. With Effective UI, you'll learn proven user-experience strategies that will satisfy your clients and customers, drive business value, and increase brand strength. This book shows you how to capture the collaborative and cooperative spirit among designers, engineers, and management required for building engaging software. You'll also learn valuable methods for maintaining focus throughout the process -- whether you're a product manager who needs a clear roadmap, a developer or designer looking for guidance and advocacy, or a businessperson who wants to understand and manage user-experience software initiatives. Learn how to build software that will: Generate engaging and interactive experiences between consumers and businesses, or between businesspeople and their information systems Account for how people work with, think about, and consume information Establish a richer means of collaboration and communication Reduce frustration by streamlining complex tasks and creating processes that are more intuitive Distinguish products, services, and brands to create a competitive advantage Create scalable systems that adapt to changing user needs and behaviors Offers comprehensive coverage of all major modeling viewpoints Provides details of collaboration and class diagrams for filling in the design-level models The Art of Type and Typography is an introduction to the art and rules of typography. Incorporating the industry standard—InDesign—for typesetting from the outset, this book serves as a guide for beginning students to learn to set type properly through tutorials, activities, and examples of student work. Encompassing the history of typography from ancient times to widespread modern use, The Art of Type and Typography provides context and fosters creativity while developing key concepts, including: The history of type; Terminology; Classification; Measurement; Spacing; Alignment; Legibility; Hierarchy; Layout and Grids; Page Elements; InDesign tools and style sheets. Writing clearly and to the point, Mary Jo Krysinski brings over 30 years of design experience to this essential guide. With a glossary, sample class activities, additional online resources and a beautiful clean design, this book is the perfect introduction for a beginning typography student, and a handy reference for those needing a refresher. This long-awaited revision of a bestseller provides a practical discussion of the nature and aims of software testing. You'll find the latest methodologies for the design of effective test cases, including information on psychological and economic principles, managerial aspects, test tools, high-order testing, code inspections, and debugging. Accessible, comprehensive, and always practical, this edition provides the key information you need to test successfully, whether a novice or a working programmer. Buy your copy today and end up with fewer bugs tomorrow. Special Features: · A LANDMARK BOOK THAT HAS ENDURED FOR 25 YEARS: With little effort on the author's part, The Art of Software Testing has continued to sell since 1978; the total sales hover just under 60,000 copies at a current price point of $140.00. Since 1988 (the earliest year that sales data is available), this book has sold 7,910 copies in special sales, and it has found its way into the college and international markets, selling 2,099 and 5,075 (through our subsidiaries) respectively· A PROLIFIC REVISION AUTHOR BEHIND A 300,000-COPY BEST-SELLER: Corey Sandler is a well-known computer title author whose book Fix Your Own PC has sold over 300,000 copies in six editions. He has held top editorial roles at a number of leading computer magazines, including Digital News, PC World and PCjr· ORIGINAL AUTHOR IS WELL-KNOWN IN THE COMPUTER WORLD: Glen Myers has a great name in the field of computing, and he is well-known for his past roles at IBM, RadiSys Corporation (which he founded and steered as CEO), and his current position as Director of Spectrum Signals· A CLASSIC PROFESSIONAL GUIDE: The Art of Software Testing is a classic guide that has won wide praise for its straightforward approach to this topic About The Book: The Art of Software Testing, Second Edition provides a practical discussion of the purpose and nature of software testing. It elucidate the latest methodologies for the design of effective test cases, and provide accessible information on psychological and economic principles, managerial aspects of testing, test tools, high-order testing, code inspections, and debugging. Are you a customer success or support executive curious about adapting industry best practices to your organization? Are you a newly-promoted customer success or support manager with plenty of ideas, but not much management experience? Or are you an executive with no hands-on experience with customer success, but wanting to learn more about how to decrease churn and improve revenue expansion from existing customers? The Art of Support is a practical guide for managers and executives that answers your

questions. In it, you will find: - Best practices for customer success and support, from designing customer lifecycle journeys, to managing day-to-day activities, to measuring results. - Nuanced recommendations to build or improve your organization. - Dozens of practical tools you can use right away such as customer scorecards, sample support portfolios, hiring checklists, decision trees for selecting support models, job ladders, and budget templates. Presents ready-to-use information on how to set up and effectively run a help desk or technical software support group. The manual provides check lists for call handling and resolving calls, determining staffing levels and cost-justifying a support center Using software prototyping techniques and the tools available in the OS/2 and Windows environments, this book shows developers how to design better user interfaces that can be used in a variety of applications, eliminating the need to "reinvent the wheel" each time a new application is required. Includes step-by-step instructions for prototyping in OS/2 and Windows. Provides a practical rather than theoretical discussion of the purpose and nature of software testing. Emphasizes methodologies for the design of effective test cases. Comprehensively covers psychological and economic principles, managerial aspects of testing, test tools, high-order testing, code inspections, and debugging. Extensive bibliography. Programmers at all levels, and programming students, will find this reference work indispensible. 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. Programming Cultures explores the relationship between software engineering and the various disciplines that benefit from new codes and programming tools. The title focuses on a range of practices including: aviation design, urban infrastructure simulation, Hollywood special effects, nanotechnology, mathematics and architecture. In terms of building design, Programming Cultures specifically examine's the potential of new software designed to solve specific visualization and data processing problems from within the profession. The book allows architects to become more familiar with programming rather than basing their work on appropriated systems designed for non-architectural applications (Maya, 3D Studio MAX etc.) and will become a primer for an emerging culture of students; academics and young professionals that are starting to outgrow the predetermined structure of today’s most popular modeling and animation packages. Software art is a practice that regards software as a cultural phenomenon that defines one of the principal domains of our existence today. Thus, software is not regarded as an invisible layer, but rather as a decisive level and a language working at reproduction of certain orders, whether aesthetic, cultural, social or political. Software art creatively questions and redefines software and its ways of functioning. This innovative book uncovers all the steps readers should follow in order to build successful software and systems With the help of numerous examples, Albin clearly shows how to incorporate Java, XML, SOAP, ebXML, and BizTalk when designing true distributed business systems Teaches how to easily integrate design patterns into software design Documents all architectures in UML and presents code in either Java or C++ This book introduces Software Thermal Management (STM) as a means of reducing power consumption in a computing system in order to manage heat, improve component reliability and increase system safety. Readers will benefit from this pragmatic guide to the field of STM for embedded systems and its catalog of software power management techniques. Since thermal management is a key bottleneck in embedded systems design, this book focuses on root cause of heat in embedded systems: power. Since software has an enormous impact on power consumption in an embedded system, this book urges software engineers to manage heat effectively by understanding, categorizing and developing new ways to reduce static and dynamic power consumption. Whereas most books on thermal management describe mechanisms to remove heat, this book focuses on ways for software engineers to avoid generating heat in the first place. The Art and Science of Analyzing Software Data provides valuable information on analysis techniques often used to derive insight from software data. This book shares best practices in the field generated by leading data scientists, collected from their experience training software engineering students and practitioners to master data science. The book covers topics such as the analysis of security data, code reviews, app stores, log files, and user telemetry, among others. It covers a wide variety of techniques such as co-change analysis, text analysis, topic analysis, and concept analysis, as well as advanced topics such as release planning and generation of source code comments. It includes stories from the trenches from expert data scientists illustrating how to apply data analysis in industry and open source, present results to stakeholders, and drive decisions. Presents best practices, hints, and tips to analyze data and apply tools in data science projects Presents research methods and case studies that have emerged over the past few years to further understanding of software data Shares stories from the trenches of successful data science initiatives in industry A guide for designing easy-to-use software, this book offers an on-the-job view of what it takes to create great products, offering practical tips and advice instead of forcing the reader to extrapolate from abstract psychological theory. "Human Interface" targets a wide range of design issues, from taming the incomprehensible interfaces of database systems and the Internet, to using sound and animation effectively in multimedia. Imagine that you are the CEO of a software company. You know you compete in an environment that does not permit you to treat innovation as a secondary issue. But how should you manage your software innovation to get the most out of it? This book will provide you with the answer. Software innovation is multifaceted and the approaches used by companies can be very different. The team of authors that wrote this book took the assumption that there is no such thing as a universal software engineering process or innovation process. Some things work well for a certain company, others do not. The book is organized around what the authors call eight fundamental practice areas for innovation with software. Each practice area contains a number of activities that can help companies to master that practice area. It also contains industrial experience reports that illustrate the applicability of these practice areas in software companies and is structured in such a way that you can select and read only those practice areas that are relevant to your company. The book is written with an industrial target audience in mind. Its most important goal is to challenge companies by offering them a framework to become more innovation-driven, rather than engineering-driven. Intrigued? Here you will find details of what you and your company can do to understand, implement, and sustain continuous innovation. The classic, landmark work on software testing The hardware and software of computing have changed markedly in the three decades since the first edition of The Art of Software Testing, but this book's powerful underlying analysis has stood the test of time. Whereas most books on software testing target particular development techniques, languages, or testing methods, The Art of Software Testing, Third Edition provides a brief but powerful and comprehensive presentation of time-proven software testing approaches. If your software development project is mission critical, this book is an investment that will pay for itself with the first bug you find. The new Third Edition explains how to apply the book's classic principles to today's hot topics including: Testing apps for iPhones, iPads, BlackBerrys, Androids, and other mobile devices Collaborative (user) programming and testing Testing for Internet applications, e-commerce, and agile programming environments Whether you're a student looking for a testing guide you'll use for the rest of your career, or an IT manager overseeing a software development team, The Art of Software Testing, Third Edition is an expensive book that will pay for itself many times over. State-of-the-Art Software Security Testing: Expert, Up to Date, and Comprehensive The Art of Software Security

Testing delivers in-depth, up-to-date, battle-tested techniques for anticipating and identifying software security problems before the “bad guys” do. Drawing on decades of experience in application and penetration testing, this book’s authors can help you transform your approach from mere “verification” to proactive “attack.” The authors begin by systematically reviewing the design and coding vulnerabilities that can arise in software, and offering realistic guidance in avoiding them. Next, they show you ways to customize software debugging tools to test the unique aspects of any program and then analyze the results to identify exploitable vulnerabilities. Coverage includes Tips on how to think the way software attackers think to strengthen your defense strategy Cost-effectively integrating security testing into your development lifecycle Using threat modeling to prioritize testing based on your top areas of risk Building testing labs for performing white-, grey-, and black-box software testing Choosing and using the right tools for each testing project Executing today’s leading attacks, from fault injection to buffer overflows Determining which flaws are most likely to be exploited by real-world attackers The nearly 60 essays in this book--always easily digestible, often profound, and never too serious--take up large themes and important questions, never shying away from controversy. (Computer Books) The Art of UNIX Programming poses the belief that understanding the unwritten UNIX engineering tradition and mastering its design patterns will help programmers of all stripes to become better programmers. This book attempts to capture the engineering wisdom and design philosophy of the UNIX, Linux, and Open Source software development community as it has evolved over the past three decades, and as it is applied today by the most experienced programmers. Eric Raymond offers the next generation of "hackers" the unique opportunity to learn the connection between UNIX philosophy and practice through careful case studies of the very best UNIX/Linux programs. The Definitive Insider’s Guide to Auditing Software Security This is one of the most detailed, sophisticated, and useful guides to software security auditing ever written. The authors are leading security consultants and researchers who have personally uncovered vulnerabilities in applications ranging from sendmail to Microsoft Exchange, Check Point VPN to Internet Explorer. Drawing on their extraordinary experience, they introduce a start-to-finish methodology for “ripping apart” applications to reveal even the most subtle and well-hidden security flaws. The Art of Software Security Assessment covers the full spectrum of software vulnerabilities in both UNIX/Linux and Windows environments. It demonstrates how to audit security in applications of all sizes and functions, including network and Web software. Moreover, it teaches using extensive examples of real code drawn from past flaws in many of the industry's highest-profile applications. Coverage includes • Code auditing: theory, practice, proven methodologies, and secrets of the trade • Bridging the gap between secure software design and post-implementation review • Performing architectural assessment: design review, threat modeling, and operational review • Identifying vulnerabilities related to memory management, data types, and malformed data • UNIX/Linux assessment: privileges, files, and processes • Windows-specific issues, including objects and the filesystem • Auditing interprocess communication, synchronization, and state • Evaluating network software: IP stacks, firewalls, and common application protocols • Auditing Web applications and technologies In any software project the analysis stage is vital to the success of the project. This book provides a thorough introduction to analysis and where it fits into the software engineering process. The author applies his many years of experience - as both a manager of software projects and as a consultant to numerous companies - to illustrate successful techniques and identify potential pitfalls. Based on courses at Columbia University for a diverse audience of students and professionals, the author is concerned throughout to emphasise the stages of analysis and to identify many alternative modelling tools that an analyst can use. Particular emphasis is placed on joint application development and on prototyping. Readers are assumed to have a reasonable understanding of computer concepts and terminology, making this suitable for a first-level analysis course or for information systems professionals who need an in-depth understanding of the principles of the analysis and design process. This book will teach you how to test computer software under real-world conditions. The authors have all been test managers and software development managers at well-known Silicon Valley software companies. Successful consumer software companies have learned how to produce high-quality products under tight time and budget constraints. The book explains the testing side of that success. Who this book is for: * Testers and Test Managers * Project Managers-Understand the timeline, depth of investigation, and quality of communication to hold testers accountable for. * Programmers-Gain insight into the sources of errors in your code, understand what tests your work will have to pass, and why testers do the things they do. * Students-Train for an entry-level position in software development. What you will learn: * How to find important bugs quickly * How to describe software errors clearly * How to create a testing plan with a minimum of paperwork * How to design and use a bug-tracking system * Where testing fits in the product development process * How to test products that will be translated into other languages * How to test for compatibility with devices, such as printers * What laws apply to software quality Offers a collection of essays on philosophies and strategies for defining, leading, and managing projects. This book explains to technical and non-technical readers alike what it takes to get through a large software or web development project. It does not cite specific methods, but focuses on philosophy and strategy. As programmers, we’ve all seen source code that’s so ugly and buggy it makes our brain ache. Over the past five years, authors Dustin Boswell and Trevor Foucher have analyzed hundreds of examples of "bad code" (much of it their own) to determine why they’re bad and how they could be improved. Their conclusion? You need to write code that minimizes the time it would take someone else to understand it—even if that someone else is you. This book focuses on basic principles and practical techniques you can apply every time you write code. Using easy-to-digest code examples from different languages, each chapter dives into a different aspect of coding, and demonstrates how you can make your code easy to understand. Simplify naming, commenting, and formatting with tips that apply to every line of code Refine your program’s loops, logic, and variables to reduce complexity and confusion Attack problems at the function level, such as reorganizing blocks of code to do one task at a time Write effective test code that is thorough and concise—as well as readable "Being aware of how the code you create affects those who look at it later is an important part of developing software. The authors did a great job in taking you through the different aspects of this challenge, explaining the details with instructive examples." —Michael Hunger, passionate Software Developer