Bookmark File PDF A Guide To Software

As recognized, adventure as capably as experience about lesson, amusement, as skillfully as contract can be gotten by just checking out a ebook **A Guide To Software** moreover it is not directly done, you could allow even more nearly this life, on the order of the world.

We have the funds for you this proper as with ease as easy pretentiousness to acquire those all. We meet the expense of A Guide To Software and numerous books collections from fictions to scientific research in any way. in the midst of them is this A Guide To Software that can be your partner.

KEY=SOFTWARE - HOPE ARIAS

Working with Coders A Guide to Software Development for the Perplexed Non-Techie *Apress* Get introduced to the fascinating world inhabited by the professional software developer. Aimed at a non-technical audience, this book aims to de-obfuscate the jargon, explain the various activities that coders undertake, and analyze the specific pressures, priorities, and preoccupations that developers are prone to. In each case it offers pragmatic advice on how to use this knowledge to make effective business decisions and work productively with software teams. Software projects are, all too often, utter nightmares for everyone involved. Depending on which study you read, between 60 and 90 percent of all software projects are completed late, run over budget, or deliver an inferior quality end product. This blight affects everyone from large organizations trying to roll out business change to tiny startups desperately trying to launch their MVP before the money runs out. While there has been much attention devoted to understanding these failings, leading to the development of entire management methodologies aimed at reducing the failure rate, such new processes have had, at best, limited success in delivering better results. Based on a decade spent exploring the world of software, Patrick Gleeson argues that the underlying reason for the high failure rate of software projects is that software development, being a deeply arcane and idiosyncratic process, tends to be thoroughly and disastrously misunderstood by managers and leaders. So long as the people tasked with making decisions about software projects are unaware of these idiosyncrasies and their ramifications, software projects will be delivered late, software products will be unfit for purpose, and relations between software developers and their non-technical colleagues will be strained. Even the most potent modern management tools are ineffective when wielded blindly. To anyone who employs, contracts, manages, or works with software developers, Working with Coders: A Guide to Software Development for the Perplexed Non-Techie delivers the understanding necessary to reduce friction and inefficiencies at the intersection between software development teams and their non-technical colleagues. What You'll Learn Discover why software projects are so commonly delivered late and with an abysmal end product Examine why the relationship between coders and their non-technical colleagues is often strained Understand how the software development process works and how to support it effectively Decipher and use the jargon of software development Keep a team of coders happy and improve the odds of successful software project delivery Who This Book Is For Anyone who employs, contracts, or manages software developers—such as tech startup CEOs, project managers, and clients of digital agencies—and wishes the relationship were easier and more productive. The secondary readership is software developers who want to find ways of working more effectively as part of a team. **Guide to Software Development Designing and Managing the Life Cycle** *Springer* This book presents a guide to navigating the complicated issues of quality and process improvement in enterprise software implementation, and the effect these have on the software development life cycle (SDLC). Offering an integrated approach that includes important management and decision practices, the text explains how to create successful automated solutions that fit user and customer needs, by mixing different SDLC methodologies. With an emphasis on the realities of practice, the book offers essential advice on defining business requirements, and managing change. This revised and expanded second edition includes new content on such areas as cybersecurity, big data, and digital transformation. Features: presents examples, case studies, and chapter-ending problems and exercises; concentrates on the skills needed to distinguish successful software implementations; considers the political and cultural realities in organizations; suggests many alternatives for how to manage and model a system. **Concise Guide to Software Engineering From Fundamentals to Application Methods** *Springer* This essential textbook presents a concise introduction to the fundamental principles of software engineering, together with practical guidance on how to apply the theory in a real-world, industrial environment. The wide-ranging coverage encompasses all areas of software design, management, and quality. Topics and features: presents a broad overview of software engineering, including software lifecycles and phases in software development, and project management for software engineering; examines the areas of requirements engineering, software configuration management, software inspections, software testing, software quality assurance, and process quality; covers topics on software metrics and problem solving, software reliability and dependability, and software design and development, including Agile approaches; explains formal methods, a set of mathematical techniques to specify and derive a program from its specification, introducing the Z specification language; discusses software process improvement, describing the CMMI model, and introduces UML, a visual modelling language for software systems; reviews a range of tools to support various activities in software engineering, and offers advice on the selection and management of a software supplier; describes such innovations in the field of software as distributed systems, service-oriented architecture, software as a service, cloud computing, and embedded systems; includes key learning topics, summaries and review questions in each chapter, together with a useful glossary. This practical and easy-to-follow textbook/reference is ideal for computer science students seeking to learn how to build high quality and reliable software on time and on budget. The text also serves as a self-study primer for software engineers, quality professionals, and software managers. **Software Engineering for Absolute Beginners Your Guide to Creating Software Products** *Apress* Start programming from scratch, no experience required. This beginners' guide to software engineering starts with a discussion of the different editors used to create software and covers setting up a Docker environment. Next, you will learn about repositories and version control along with its uses. Now that you are ready to program, you'll go through the basics of Python, the ideal language to learn as a novice software engineer. Many modern applications need to talk to a database of some kind, so you will explore how to create and connect to a database and how to design one for your app. Additionally you will discover how to use Python's Flask microframework and how to efficiently test your code. Finally, the book explains best practices in coding, design, deployment, and security. Software Engineering for Absolute Beginners answers the question of what topics you should know when you start out to learn software engineering. This book covers a lot of topics, and aims to clarify the hidden, but very important, portions of the software development toolkit. After reading this book, you, a complete beginner, will be able to identify best practices and efficient approaches to software development. You will be able to go into a work environment and recognize the technology and approaches used, and set up a professional environment to create your own software applications. What You Will Learn Explore the concepts that you will encounter in the majority of companies doing software development Create readable code that is neat as well as well-designed Build code that is source controlled, containerized, and deployable Secure your codebase Optimize your workspace Who This Book Is For A reader with a keen interest in creating software. It is also helpful for students. **Software Design A Comprehensive Guide to Software Development Projects** *CRC Press* This book is perhaps the first attempt to give full treatment to the topic of Software Design. It will facilitate the academia as well as the industry. This book covers all the topics of software design including the ancillary ones. **Software Metrics A Guide to Planning, Analysis, and Application** *CRC Press* The modern field of software metrics emerged from the computer modeling and "statistical thinking" services of the 1980s. As the field evolved, metrics programs were integrated with project management, and metrics grew to be a major tool in the managerial decision-making process of software companies. Now practitioners in the software industry have **Building Software A Practitioner's Guide** *CRC Press* Novel in its approach to software design, development, and management, Building Software: A Practitioner's Guide shows you how to successfully build and manage a system. The approach the authors recommend is a simple, effective framework known as Solution Engineering Execution (SEE). Through SEE, you create a successful solution by following a high **Team Guide to Software Testability: Better Software Through Greater Testability** *Conflux Books* Testability is a vital property of modern software. It enables software teams to make changes rapidly and safely with clear feedback loops to understand the impact of changes. When your product is testable, it is more likely to meet all of your customer's needs. If you want to drive improvements in both speed and agility, testability is the fuel you need to deliver modern software. **The Complete Software Developer's Career Guide** *Simple Programmer, LLC* "Early in his software developer career, John Sonmez discovered that technical knowledge alone isn't enough to break through to the next income level - developers need "soft skills" like the ability to learn new technologies just in time, communicate clearly with management and consulting clients, negotiate a fair hourly rate, and unite teammates and coworkers in working toward a common goal. Today John helps more than 1.4 million programmers every year to increase their income by developing this unique blend of skills. Who Should Read This Book? Entry-Level Developers - This book will show you how to ensure you have the technical skills your future boss is looking for, create a resume that leaps off a hiring manager's desk, and escape the "no work experience" trap. Mid-Career Developers - You'll see how to find and fill in gaps in your technical knowledge, position yourself as the one team member your boss can't live without, and turn those dreaded annual reviews into chance to make an iron-clad case for your salary bump. Senior Developers - This book will show you how to become a specialist who can command above-market wages, how building a name for yourself can make opportunities come to you, and how to decide whether consulting or entrepreneurship are paths you should pursue. Brand New Developers - In this book you'll discover what it's like to be a professional software developer, how to go from "I know some code" to possessing the skills to work on a development team, how to speed along your learning by avoiding common beginner traps, and how to decide whether you should invest in a programming degree or 'bootcamp.'"-- **The Complete Guide to Software Testing** **A Wiley-QED Publication* The Complete Guide to Software Testing Bill Hetzel Gain a new perspective to software testing as a life cycle activity, not merely as something that happens at the end of coding. This edition is completely revised and contains new chapters on testing methodologies including ANSI standard-based testing—a survey of testing practices. Dr. Hetzel first develops the concepts and principles of testing. Then he presents detailed discussions of testing techniques, methodologies and management perspectives. Each chapter contains examples, checklists and case studies based on Dr. Hetzel's consulting and management experience. These will help you understand the material and adapt it to your environment. Intended primarily for software developers, testers and managers, auditors and quality assurance specialists will find the book an invaluable aid for the development of testing standards and the evaluation of testing effectiveness. Table of Contents: Introduction. Principles of Testing. Methodology. Testing through Reviews. Testing Requirements. Testing Design. Testing Programs—Testing in the Small. Testing Systems—Testing in the Large. Testing Software Changes. Testing Software Packages. The Role of Management. Organizing the Testing Function. Controlling the Testing Function. Putting the Pieces Together. Testing Practices Survey. Sample Testing Policies. Quality Measurement Diagnostic Checklist. Testing References (Bibliography). **The Complete Guide to Software As a Service Everything You Need to Know About Saas** *Createspace Independent Publishing Platform* The Complete Guide to Software as a Service is truly "everything you need to know about SaaS." This is the only book available today that covers the multiple facets of the SaaS model: functional, operational, technical, security and financial. Starting with the definition and the origins of SaaS, it gives a 360-degree view of the inner workings of a SaaS business. This book is a must read for entrepreneurs who are launching a SaaS company. Learn the six ways to fail your SaaS start-up. It will also guide any software company who is transitioning from an on-premise license model to SaaS. Learn what IT and business functions must evolve when moving from one business model to the next. It also provides useful information and insight to different functional managers within a SaaS company. As well, users of SaaS software will become more knowledgeable clients of their SaaS providers after reading this book. Learn how to "read between the lines" of your SaaS contract and focus on the clauses where you have real negotiating power. For anyone interested in learning more about this important shift in the software industry, this book fills a void that exists today in the world of SaaS. **The Complete Guide to Software Testing** Ed Yourdan called it a bible for project managers. You'll gain a new perspective on software testing as a life cycle activity, not merely as something that happens at the end of coding. An invaluable aid for the development of testing standards and the evaluation of testing effectiveness. **Guide to Efficient Software Design An MVC Approach to Concepts, Structures, and Models** *Springer Nature* This classroom-tested textbook presents an active-learning approach to the foundational concepts of software design. These concepts are then applied to a case study, and reinforced through practice exercises, with the option to follow either a structured design or object-oriented design paradigm. The text applies an incremental and iterative software development approach, emphasizing the use of design characteristics and modeling techniques as a way to represent higher levels of design abstraction, and promoting the model-view-controller (MVC) architecture. Topics and features: provides a case study to illustrate the various concepts discussed throughout the book, offering an in-depth look at the pros and cons of different software designs; includes discussion questions and hands-on exercises that extend the case study and apply the concepts to other problem domains; presents a review of program design fundamentals to reinforce understanding of the basic concepts; focuses on a bottom-up approach to describing software design concepts; introduces the characteristics of a good software design, emphasizing the model-view-controller as an underlying architectural principle; describes software design from both object-oriented and structured perspectives; examines additional topics on human-computer interaction design, quality assurance, secure design, design patterns, and persistent data storage design; discusses design concepts that may be applied to many types of software development projects; suggests a template for a software design document, and offers ideas for further learning. Students of computer science and software engineering will find

this textbook to be indispensable for advanced undergraduate courses on programming and software design. Prior background knowledge and experience of programming is required, but familiarity in software design is not assumed. **Guide to Advanced Empirical Software Engineering** *Springer Science & Business Media* This book gathers chapters from some of the top international empirical software engineering researchers focusing on the practical knowledge necessary for conducting, reporting and using empirical methods in software engineering. Topics and features include guidance on how to design, conduct and report empirical studies. The volume also provides information across a range of techniques, methods and qualitative and quantitative issues to help build a toolkit applicable to the diverse software development contexts **The Definitive Guide to Free Software** Free software always sets off alarm bells. Will it be as good as paid-for packages? Will it try to sneak junk on to your PC? Will it constantly nag you to update to a premium version? This guide, written by the experts at Computeractive and Web User, scours the internet to find the very best free programs that deliver high-quality features and no nasty surprises. **A Practitioner's Guide to Software Test Design** *Artech House* Written by a leading expert in the field, this unique volume contains current test design approaches and focuses only on software test design. Copeland illustrates each test design through detailed examples and step-by-step instructions. **A Guide to Software Package Evaluation & Selection The R2ISC Method** *Amacom Books* Offers a method for evaluating a business software package against five criteria--current requirements, future requirements, ease of implementation, vendor support, and cost. The CD-ROM contains a sample request for proposal to send vendors and project plan. Annotation copyrighted by Book News, Inc., Portland, OR **The Software Development Lifecycle - A Complete Guide** *Richard Murch* This book provides a step by step guide to all the processes, goals, inputs, outputs and many other aspects of a repeatable software methodology for ANY project. From “soup to nuts” ... the whole shebang ~! All in one place at an incredible price.... over 130 pages of knowledge. Any information technology organization must have a highly structured framework into which it can place processes, principles, and guidelines. The framework used for software development is a called a lifecycle. The software development lifecycle (SDLC) defines a repeatable process for building information system that incorporate guidelines, methodologies, and standards. A lifecycle delivers value to an organization by addressing specific business needs within the software application development environment. The implementation of a lifecycle aids project managers in minimizing system development risks, eliminating redundancy, and increasing efficiencies. It also encourages reuse, redesign, and, more importantly, reducing costs. **Guide To Software Export: A Handbook For International Software Sales** *Routledge* An ideal reference source for CEOs, marketing and sales managers, sales consultants, and students of international marketing, Guide to Software Export provides a step-by-step approach to initiating or expanding international software sales. It teaches you how to examine critically your candidate product for exportability; how to find distributors, agents, and resellers abroad; how to identify the best distribution structure for export; and much, much more! Not content with providing just the guidelines for setting up, expanding, and managing your international sales channels, Guide to Software Export advises you on pitfalls to avoid, important legal and financial considerations associated with software export, and essential market and distribution information. In an effort to cover all the bases, this comprehensive text also discusses: negotiating partnerships electronic marketing evaluating the competition cultural assumptions and biases adapting software for use in Asian markets information sources on the Internet distribution channel strategies If you’re not satisfied with your company’s international sales performance or you want to get into the global market, Guide to Software Export can help you guide your company through the transition. With the book’s easy-to-follow advice and checkpoints, you are sure to bring new levels of success to your company, so act now and get out in the forefront of software exporting. **Global Software and IT A Guide to Distributed Development, Projects, and Outsourcing** *John Wiley & Sons* Based on the author’s first-hand experience and expertise,this book offers a proven framework for global softwareengineering. Readers will learn best practices for managing avariety of software projects, coordinating the activities ofseveral locations across the globe while accounting for culturaldifferences. Most importantly, readers will learn how to engineer afirst-rate software product as efficiently as possible by fullyleveraging global personnel and resources. Global Software and IT takes a unique approach that works forprojects of any size, examining such critical topics as: Executing a seamless project across multiple locations Mitigating the risks of off-shoring Developing and implementing processes for globaldevelopment Establishing practical outsourcing guidelines Fostering effective collaboration and communication acrosscontinents and culture This book provides a balanced framework for planning globaldevelopment, covering topics such as managing people in distributedsites and managing a project across locations. It delivers acomprehensive business model that is beneficial to anyone lookingfor the most cost-effective, efficient way to engineer goodsoftware products. **Concise Guide to Software Testing** *Springer Nature* This practically-focused textbook provides a concise and accessible introduction to the field of software testing, explaining the fundamental principles and offering guidance on applying the theory in an industrial environment. Topics and features: presents a brief history of software quality and its influential pioneers, as well as a discussion of the various software lifecycles used in software development; describes the fundamentals of testing in traditional software engineering, and the role that static testing plays in building quality into a product; explains the process of software test planning, test analysis and design, and test management; discusses test outsourcing, and test metrics and problem solving; reviews the tools available to support software testing activities, and the benefits of a software process improvement initiative; examines testing in the Agile world, and the verification of safety critical systems; considers the legal and ethical aspects of software testing, and the importance of software configuration management; provides key learning topics and review questions in every chapter, and supplies a helpful glossary at the end of the book. This easy-to-follow guide is an essential resource for undergraduate students of computer science seeking to learn about software testing, and how to build high quality and reliable software on time and on budget. The work will also be of interest to industrialists including software engineers, software testers, quality professionals and software managers, as well as the motivated general reader. **The Missing README A Guide for the New Software Engineer** *No Starch Press* Key concepts and best practices for new software engineers — stuff critical to your workplace success that you weren’t taught in school. For new software engineers, knowing how to program is only half the battle. You’ll quickly find that many of the skills and processes key to your success are not taught in any school or bootcamp. The Missing README fills in that gap—a distillation of workplace lessons, best practices, and engineering fundamentals that the authors have taught rookie developers at top companies for more than a decade. Early chapters explain what to expect when you begin your career at a company. The book’s middle section expands your technical education, teaching you how to work with existing codebases, address and prevent technical debt, write production-grade software, manage dependencies, test effectively, do code reviews, safely deploy software, design evolvable architectures, and handle incidents when you’re on-call. Additional chapters cover planning and interpersonal skills such as Agile planning, working effectively with your manager, and growing to senior levels and beyond. You’ll learn: • How to use the legacy code change algorithm, and leave code cleaner than you found it • How to write operable code with logging, metrics, configuration, and defensive programming • How to write deterministic tests, submit code reviews, and give feedback on other people’s code • The technical design process, including experiments, problem definition, documentation, and collaboration • What to do when you are on-call, and how to navigate production incidents • Architectural techniques that make code change easier • Agile development practices like sprint planning, stand-ups, and retrospectives This is the book your tech lead wishes every new engineer would read before they start. By the end, you’ll know what it takes to transition into the workplace–from CS classes or bootcamps to professional software engineering. **Inroads to Software Quality "how To" Guide and Toolkit** *Prentice Hall* Helps software organizations build in quality cost-effectively, starting before products are developed. This book is a highly-readable, non-theoretical guide to software quality improvement. It includes 18 "filters" that software development managers can use to instill quality throughout the development process. Presents techniques that can lead to a dramatic reduction in expensive, time-consuming functional testing. Covers all the leading process improvement tools. Managers responsible for quality processes, directors of R&D, development engineers, software testers and QA managers, process improvement engineers, business and engineering faculty, corporate trainers and ISO 9000 implementors. **How to Break Software A Practical Guide to Testing** *Pearson* CD-ROM contains: Canned HEAT v.2.0 -- Holodeck Lite v. 1.0. **Software Methodologies A Quantitative Guide** *CRC Press* This comprehensive reference uses a formal and standard evaluation technique to show the strengths and weakness of more than 60 software development methodologies such as agile, DevOps, RUP, Waterfall, TSP, XP and many more. Each methodology is applied to an application of 1000 function points using the Java language. Each methodology produces a characteristic set of results for development schedules, productivity, costs, and quality. The intent of the book is to show readers the optimum kinds of methodologies for the projects they are concerned with and to warn them about counter indications and possible harm from unsuitable methodologies. **Building a Career in Software A Comprehensive Guide to Success in the Software Industry** *Apress* Software engineering education has a problem: universities and bootcamps teach aspiring engineers to write code, but they leave graduates to teach themselves the countless supporting tools required to thrive in real software companies. Building a Career in Software is the solution, a comprehensive guide to the essential skills that instructors don't need and professionals never think to teach: landing jobs, choosing teams and projects, asking good questions, running meetings, going on-call, debugging production problems, technical writing, making the most of a mentor, and much more. In over a decade building software at companies such as Apple and Uber, Daniel Heller has mentored and managed tens of engineers from a variety of training backgrounds, and those engineers inspired this book with their hundreds of questions about career issues and day-to-day problems. Designed for either random access or cover-to-cover reading, it offers concise treatments of virtually every non-technical challenge you will face in the first five years of your career—as well as a selection of industry-focused technical topics rarely covered in training. Whatever your education or technical specialty, Building a Career in Software can save you years of trial and error and help you succeed as a real-world software professional. What You Will Learn Discover every important nontechnical facet of professional programming as well as several key technical practices essential to the transition from student to professional Build relationships with your employer Improve your communication, including technical writing, asking good questions, and public speaking Who This Book is For Software engineers either early in their careers or about to transition to the professional world; that is, all graduates of computer science or software engineering university programs and all software engineering boot camp participants. **Testing in Scrum A Guide for Software Quality Assurance in the Agile World** *Rocky Nook, Inc.* These days, more and more software development projects are being carried out using agile methods like Scrum. Agile software development promises higher software quality, a shorter time to market, and improved focus on customer needs. However, the transition to working within an agile methodology is not easy. Familiar processes and procedures change drastically. Software testing and software quality assurance have a crucial role in ensuring that a software development team, department, or company successfully implements long-term agile development methods and benefits from this framework. This book discusses agile methodology from the perspective of software testing and software quality assurance management. Software development managers, project managers, and quality assurance managers will obtain tips and tricks on how to organize testing and assure quality so that agile projects maintain their impact. Professional certified testers and software quality assurance experts will learn how to work successfully within agile software teams and how best to integrate their expertise. Topics include: Agile methodology and classic process models How to plan an agile project Unit tests and test first approach Integration testing and continuous integration System testing and test nonstop Quality management and quality assurance Also included are five case studies from the manufacturing, online-trade, and software industry as well as test exercises for self-assessment. This book covers the new ISTQB Syllabus for Agile Software Testing and is a relevant resource for all students and trainees worldwide who plan to undertake this ISTQB certification. **A+ Guide to IT Technical Support** *Cengage Learning* Master the details of IT technical support as Andrews/Dark/West's comprehensive COMPTIA A+ GUIDE TO IT TECHNICAL SUPPORT, 10E explains how to work with users as well as install, maintain, troubleshoot and network computer hardware and software. This step-by-step, highly visual, best-selling approach uses CompTIA A+ Exam objectives as a framework to prepare you for 220-1001 and 220-1002 certification exams. Each chapter covers core and advanced topics while emphasizing practical application of the most current technology, techniques and industry standards. You study the latest hardware, security, Active Directory, operational procedures, basics of scripting, virtualization, cloud computing, mobile devices and Windows 10. Lab Manuals, CourseNotes, online labs and optional MindTap online resources provide additional certification test preparation and interactive activities to prepare you for a role as an IT support technician or administrator. **ARM System Developer's Guide Designing and Optimizing System Software** *Elsevier* Over the last ten years, the ARM architecture has become one of the most pervasive architectures in the world, with more than 2 billion ARM-based processors embedded in products ranging from cell phones to automotive braking systems. A world-wide community of ARM developers in semiconductor and product design companies includes software developers, system designers and hardware engineers. To date no book has directly addressed their need to develop the system and software for an ARM-based system. This text fills that gap. This book provides a comprehensive description of the operation of the ARM core from a developer’s perspective with a clear emphasis on software. It demonstrates not only how to write efficient ARM software in C and assembly but also how to optimize code. Example code throughout the book can be integrated into commercial products or used as templates to enable quick creation of productive software. The book covers both the ARM and Thumb instruction sets, covers Intel's XScale Processors, outlines distinctions among the versions of the ARM architecture, demonstrates how to implement DSP algorithms, explains exception and interrupt handling, describes the cache technologies that surround the ARM cores as well as the most efficient memory management techniques. A final chapter looks forward to the future of the ARM architecture considering ARMv6, the latest change to the instruction set, which has been designed to improve the DSP and media processing capabilities of the architecture. * No other book describes the ARM core from a system and software perspective. * Author team combines extensive ARM software engineering experience with an in-depth knowledge of ARM developer needs. * Practical, executable code is fully explained in the book and available on the publisher's Website. * Includes a simple embedded operating system. **Guide to the Software Engineering Body of Knowledge (Swebok(r)) Version 3.0** In the Guide to the Software Engineering Body of Knowledge (SWEBOK(R) Guide), the IEEE Computer Society establishes a baseline for the body of knowledge for the field of software engineering, and the work supports the Society's responsibility to promote the advancement of both theory

and practice in this field. It should be noted that the Guide does not purport to define the body of knowledge but rather to serve as a compendium and guide to the knowledge that has been developing and evolving over the past four decades. Now in Version 3.0, the Guide's 15 knowledge areas summarize generally accepted topics and list references for detailed information. The editors for Version 3.0 of the SWEBOK(R) Guide are Pierre Bourque (Ecole de technologie superieure (ETS), Universite du Quebec) and Richard E. (Dick) Fairley (Software and Systems Engineering Associates (S2EA)). **Write Portable Code An Introduction to Developing Software for Multiple Platforms** *No Starch Press* Contains lessons on cross-platform software development, covering such topics as portability techniques, source control, compilers, user interfaces, and scripting languages. **Software Security Engineering A Guide for Project Managers** *Addison-Wesley Professional* Software Security Engineering draws extensively on the systematic approach developed for the Build Security In (BSI) Web site. Sponsored by the Department of Homeland Security Software Assurance Program, the BSI site offers a host of tools, guidelines, rules, principles, and other resources to help project managers address security issues in every phase of the software development life cycle (SDLC). The book's expert authors, themselves frequent contributors to the BSI site, represent two well-known resources in the security world: the CERT Program at the Software Engineering Institute (SEI) and Cigital, Inc., a consulting firm specializing in software security. This book will help you understand why Software security is about more than just eliminating vulnerabilities and conducting penetration tests Network security mechanisms and IT infrastructure security services do not sufficiently protect application software from security risks Software security initiatives should follow a risk-management approach to identify priorities and to define what is "good enough"–understanding that software security risks will change throughout the SDLC Project managers and software engineers need to learn to think like an attacker in order to address the range of functions that software should not do, and how software can better resist, tolerate, and recover when under attack **Software Project Management Methods and Techniques** A comprehensive treatment of the practice of Software Project Management incorporating dozens of methods and demonstrating to the reader how and when to use them. In the words of one academic, "An easy to read treatment of a challenging subject." The text relfects the broad background and sucesses of its author. Included are discussions of how to recover a challenged project. Topics include how to build a team, optimization of projectsp lans, risk management, earned value management, lean development, Agile and other styles of development, the Balanced Scorecard and methods for evaluating software professionals. **Software Project Survival Guide** *Irwin/McGraw-Hill* How to make sure your next important project isn't your last.Equip yourself with SOFTWARE PROJECT SURVIVAL GUIDE. It's for everyone with a stake in the outcome of a development project--and especially for those without formal software project management training. That includes top managers, executives, clients, investors, end-user representatives, project managers, and technical leads.Here you'll find guidance from the acclaimed author of the classics CODE COMPLETE and RAPID DEVELOPMENT. Steve McConnell draws on solid research and a career's worth of hard-won experience to map the surest path to your goal--what he calls one specific approach to software development that works pretty well most of the time for most projects. Nineteen chapters in four sections cover the concepts and strategies you need for mastering the development process, including planning, design, management, quality assurance, testing, and archiving. For newcomers and seasoned project managers alike, SOFTWARE PROJECT SURVIVAL GUIDE draws on a vast store of techniques to create an elegantly simplified and reliable framework for project management success.So don't worry about wandering among complex sets of project management techniques that require years to sort out and master. SOFTWARE PROJECT SURVIVAL GUIDE goes straight to the heart of the matter to help your projects succeed. And that makes it a required addition to every professional's bookshelf. **Guide to Advanced Software Testing** *Artech House* The book offers you a practical understanding of essential software testing topics and their relationships and interdependencies. This unique resource provides a thorough overview of software testing and its purpose and value. It covers topics ranging from handling failures, faults, and mistakes, to the cost of fault corrections, OC scopingOCO the test effort and using standards to guide testing." **Team Geek A Software Developer's Guide to Working Well with Others** "*O'Reilly Media, Inc.*" In a perfect world, software engineers who produce the best code are the most successful. But in our perfectly messy world, success also depends on how you work with people to get your job done. In this highly entertaining book, Brian Fitzpatrick and Ben Collins-Sussman cover basic patterns and anti-patterns for working with other people, teams, and users while trying to develop software. This is valuable information from two respected software engineers whose popular series of talks—including "Working with Poisonous People"—has attracted hundreds of thousands of followers. Writing software is a team sport, and human factors have as much influence on the outcome as technical factors. Even if you've spent decades learning the technical side of programming, this book teaches you about the often-overlooked human component. By learning to collaborate and investing in the "soft skills" of software engineering, you can have a much greater impact for the same amount of effort. Team Geek was named as a Finalist in the 2013 Jolt Awards from Dr. Dobb's Journal. The publication's panel of judges chose five notable books, published during a 12-month period ending June 30, that every serious programmer should read. **Writing Scientific Software A Guide to Good Style** *Cambridge University Press* The core of scientific computing is designing, writing, testing, debugging and modifying numerical software for application to a vast range of areas: from graphics, meteorology and chemistry to engineering, biology and finance. Scientists, engineers and computer scientists need to write good code, for speed, clarity, flexibility and ease of re-use. Oliveira and Stewart's style guide for numerical software points out good practices to follow, and pitfalls to avoid. By following their advice, readers will learn how to write efficient software, and how to test it for bugs, accuracy and performance. Techniques are explained with a variety of programming languages, and illustrated with two extensive design examples, one in Fortran 90 and one in C++: other examples in C, C++, Fortran 90 and Java are scattered throughout the book. This manual of scientific computing style will be an essential addition to the bookshelf and lab of everyone who writes numerical software. **Designing Secure Software A Guide for Developers** *No Starch Press* What every software professional should know about security. Designing Secure Software consolidates Loren Kohnfelder's more than twenty years of experience into a concise, elegant guide to improving the security of technology products. Written for a wide range of software professionals, it emphasizes building security into software design early and involving the entire team in the process. The book begins with a discussion of core concepts like trust, threats, mitigation, secure design patterns, and cryptography. The second part, perhaps this book's most unique and important contribution to the field, covers the process of designing and reviewing a software design with security considerations in mind. The final section details the most common coding flaws that create vulnerabilities, making copious use of code snippets written in C and Python to illustrate implementation vulnerabilities. You'll learn how to: • Identify important assets, the attack surface, and the trust boundaries in a system • Evaluate the effectiveness of various threat mitigation candidates • Work with well-known secure coding patterns and libraries • Understand and prevent vulnerabilities like XSS and CSRF, memory flaws, and more • Use security testing to proactively identify vulnerabilities introduced into code • Review a software design for security flaws effectively and without judgment Kohnfelder's career, spanning decades at Microsoft and Google, introduced numerous software security initiatives, including the co-creation of the STRIDE threat modeling framework used widely today. This book is a modern, pragmatic consolidation of his best practices, insights, and ideas about the future of software. **Software Leadership A Guide to Successful Software Development** *Addison-Wesley Professional* Software and project management consultant Murray Cantor discusses how to be a good manager and how to build a competitive software team. The text is intended to be accessible to managers with little software background as well as those with extensive experience. A sampling of topics includes software architecture, developing products, improving the efficiency of the organization, the Rational Unified Process, and team leadership. c. Book News Inc. **Guide to Software Systems Development Connecting Novel Theory and Current Practice** *Springer* This book argues that the key problems of software systems development (SSD) are socio-technical rather than purely technical in nature. Software systems are unique. They are the only human artefacts that are both intangible and determinant. This presents unprecedented problems for the development process both in determining what is required and how it is developed. Primarily this is a problem of communications between stakeholders and developers, and of communications within the development team. Current solutions are not only inadequate in expressing the technical problem, they also evade the communications problems almost entirely. Whilst the book addresses the theoretical aspects of the process, its fundamental philosophy is anchored in the practical problems of everyday software development. It therefore offers both a better understanding of the problems of SSD and practical suggestions of how to deal with those problems. It is intended as a guide for practising IT project managers, particularly those who are relatively new to the position or do not have a strong IT development background. The book will also benefit students in computing and computer-related disciplines who need to know how to develop high quality systems. Software systems development (particularly of large projects) has a notoriously poor track record of delivering projects on time, on budget, and of meeting user needs. Proponents of software engineering suggest that this is because too few project managers actually comply with the disciplines demanded of the process. It is time to ask the question, if this is the case, why might this be? Perhaps instead, it is not the project managers who are wrong, but the definition of the process. The new understanding of the SSD presented here offers alternative models that can help project managers address the difficulties they face and better achieve the targets they are set. This book argues that time is up for the software engineering paradigm of SSD and that it should be replaced with a socio-technical paradigm based on open systems thinking.