Modern Software Sales Engineering It Isnt All Just Ping Pong And Beer

Professionals in the interdisciplinary field of computer science focus on the design, operation, and maintenance of computational systems and software. Methodologies and tools of engineering are utilized alongside computer applications to develop efficient and precise information databases. Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications is a comprehensive reference source for the latest scholarly material on trends, techniques, and uses of various technology applications and examines the benefits and challenges of these computational developments. Highlighting a range of pertinent topics such as utility computing, computer security, and information systems applications, this multi-volume book is ideally designed for academicians, researchers, students, web designers, software developers, and practitioners interested in computer systems and software engineering.

A new approach to safety, based on systems thinking, that is more effective, less costly, and easier to use than current techniques. Engineering has experienced a technological revolution, but the basic engineering techniques applied in safety and reliability engineering, created in a simpler, analog world, have changed very little over the years. In this groundbreaking book, Nancy Leveson proposes a new approach to safety—more suited to today's complex, sociotechnical, software-intensive world—based on modern systems thinking and systems theory. Revisiting and updating ideas pioneered by 1950s aerospace engineers in their System Safety concept, and testing her new model extensively on real-world examples, Leveson has created a new approach to safety that is more effective, less expensive, and easier to use than current techniques. Arguing that traditional models of causality are inadequate, Leveson presents a new, extended model of causation (Systems-Theoretic Accident Model and Processes, or STAMP), then shows how the new model can be used to create techniques for system safety engineering, including accident analysis, hazard analysis, system design, safety in operations, and management of safety-critical systems. She applies the new techniques to real-world events including the friendly-fire loss of a U.S. Blackhawk helicopter in the first Gulf War; the Vioxx recall; the U.S. Navy SUBSAFE program; and the bacterial contamination of a public water supply in a Canadian town. Leveson's approach is relevant even beyond safety engineering, offering techniques for “reengineering” any large sociotechnical system to improve safety and manage risk.

Have you ever seen a bad software demo ? Peter Cohan helps organizations put the Wow! into their demos to make them crisp, compelling and successful - to get the job done. He has had roles in four corners: technical, product and field marketing (he was banished to Basel, Switzerland for two years for bad behavior); sales and sales management; senior management (he built a business unit up from an empty spreadsheet into a $30M per year operation); and, in this last role, he has been that most important of all possible entities, a customer Peter Cohan leverages twenty-five years of experience in selling and marketing business software and as a customer. The Great Demo! method comes directly from extensive firsthand experiences in developing and delivering software demonstrations, and in coaching others to achieve surprisingly high success rates with their sales and marketing demos. For more information on demonstration methods, guidelines and tips, explore the author's website at www.SecondDerivative.com or contact the author directly at PCohan@SecondDerivative.com.

For more than 40 years, Computerworld has been the leading source of technology news and information for IT influencers worldwide. Computerworld's award-winning Web site (Computerworld.com), twice-monthly publication, focused conference series and custom research form the hub of the world's largest global IT media network.

No more than today, in the era of cloud technologies and social distancing, could the old saying "people buy from people" be more appropriate. Demo Guru focuses on bringing the human aspect back into the world of technical sales by establishing a perfect connection between Sales, Presales, and Prospective Customers as a key driver to unbeatable win rates. Profiting from years of experience in demonstrating enterprise software across the globe, this handbook is the Holy Grail for any experienced or novice Sales Engineer who passionately takes pride in evangelizing software solutions. Demo Guru provides all the essential tools to master the Presales profession to excellence. Provocative case studies, factual tips, and humorous true stories from the fields navigate best practices and new trends with the immutable goal of establishing Presales consultants as the trusted side of any sales process. From soft skills development to engaging audience interactions, this guide offers insightful information and innovative techniques necessary to excel at the most typical day-in-the-life Presales activities, including RFP responses, web demonstrations, and road-show demo marathons. It also provides intriguing insights on how to evolve the traditional Presales experience to serve the needs of Product Management, Marketing, R&D, and Sales Enablement. Demo Guru is a testament to the highly rewarding profession of Sales Engineering for any consultative sales fanatic and the critical function it represents for any software organization.

Software engineering research can trace its roots to a few highly influential individuals. Among that select group is Leon J. Osterweil, who has been a major force in driving software engineering from its infancy to its modern reality. For more than three decades, Prof. Osterweil's work has fundamentally defined or significantly impacted major directions in software analysis, development tools and environments, and software process--all critical parts of software engineering as it is practiced today. His exceptional contributions to the field have been recognized with numerous awards and honors through his career, including the ACM SIGSOFT Outstanding Research Award, in recognition of his extensive and sustained research impact, and the ACM SIGSOFT Influential Educator Award, in recognition of his career-long achievements as an educator and mentor. In honor of Prof. Osterweil's profound accomplishments, this book was prepared for a special honorary event held during the 2011 International Conference on Software Engineering (ICSE). It contains some of his most important published works to date, together with several new articles written by leading authorities in the field, exploring the

broad impact of his work in the past and how it will further impact software engineering research in the future. These papers, part of the core software engineering legacy and now available in one commented volume for the first time, are grouped into three sections: flow analysis for software dependability, the software lifecycle, and software process.
Enable Your Buyer for Faster B2B Sales Garin Hess, the founder and CEO of Consensus, the leader in intelligent demo automation software, points out that when it comes to B2B sales effectiveness, the real challenge for salespeople is to get better at understanding and facilitating their customers' buying group and buying process. Sales teams can shorten sales cycles and increase close rates by learning to equip their champion—the people promoting their solution inside the target account—effectively by using the DEEP-C™ buyer enablement framework: Discover, Equip, Engage, Personalize, and Coach. This book guides sales leaders and professionals through the process of moving from a sales-focused approach to a buyer enablement model.

TECHNICAL SALES ENGINEERS / TECHNICAL PRESALES SUPPORT: In today's digital economy, software is eating the world, and the companies with the best sales demonstrations are winning the game. Is a convincing demonstration the only thing that's standing between you and your next customer? Are you ready to make your next demo the best demo of the year? Do you feel that you can do better but don't know how? NEVER AGAIN LOSE A DEAL YOU SHOULD HAVE WON! Walk into ever demo feeling confident and prepared Include the one critical moment that must be in every demo Hit that home run and know how to set it up Master the art of answering difficult questions Leverage the power of saying NO with ease A BOOK WRITTEN SPECIFICALLY FOR YOU! Avoid late nights and long sales cycles Accelerate pipeline velocity and close more deals Learn and apply the best practices in the business Know exactly what to say and do before, during and after a demo Achieve the technical win alarming, predictable consistency This book addresses the root causes of the most common mistakes made by sales engineers. Add it to your cart NOW to permanently improve your software demos and sales results.

A guide to being a Software Sales Engineer in the modern world. Get started understanding the tenets of Sales Engineering, hiring SEs, and giving the best presentations possible. This book includes a breakdown on different types of Sales Engineers, tips and tricks on presentations, and humorous stories to help every SE and manager on their way to greatness.

This indispensable sales tool shows you the ropes of lead qualification, the RFP process, and needs analysis and discovery, and explains how your technical know-how can add invaluable leverage to sales efforts at every step. You learn how to plan and present the perfect pitch, demonstrate products effectively, build customer relationship skills, handle objections and competitors, negotiate prices and contracts, close the sale, and so much more - including how to avoid the critical selling mistakes so often made by technical pros who jump to sales. The book also addresses key career management and team-building topics, and includes detailed case studies, concise chapter summaries, and handy checklists of skill-building tips that reinforce all the career-boosting skills and techniques you learn.

A modern and unified treatment of the mechanics, planning, and control of robots, suitable for a first course in robotics.

Jeff Lawson, software developer turned CEO of Twilio, creates a new playbook for unleashing the full potential of software developers in any organization, showing how to help management utilize this coveted and valuable workforce to enable growth, solve a wide range of business problems and drive digital transformation. From banking and retail to insurance and finance, every industry is turning digital, and every company needs the best software to win the hearts and minds of customers. The landscape has shifted from the classic build vs. buy question, to one of build vs. die. Companies have to get this right to survive. But how do they make this transition? Software developers are sought after, highly paid, and desperately needed to compete in the modern, digital economy. Yet most companies treat them like digital factory workers without really understanding how to unleash their full potential. Lawson argues that developers are the creative workforce who can solve major business problems and create hit products for customers—not just grind through rote tasks. From Google and Amazon, to one-person online software companies—companies that bring software developers in as partners are winning. Lawson shows how leaders who build industry changing software products consistently do three things well. First, they understand why software developers matter more than ever. Second, they understand developers and know how to motivate them. And third, they invest in their developers' success. As a software developer and public company CEO, Lawson uses his unique position to bridge the language and tools executives use with the unique culture of high performing, creative software developers. Ask Your Developer is a toolkit to help business leaders, product managers, technical leaders, software developers, and executives achieve their common goal—building great digital products and experiences. How to compete in the digital economy? In short: Ask Your Developer.

SEMAT (Software Engineering Methods and Theory) is an international initiative designed to identify a common ground, or universal standard, for software engineering. It is supported by some of the most distinguished contributors to the field. Creating a simple language to describe methods and practices, the SEMAT team expresses this common ground as a kernel–or framework–of elements essential to all software development. The Essence of Software Engineering introduces this kernel and shows how to apply it when developing software and improving a team's way of working. It is a book for software professionals, not methodologists. Its usefulness to development team members, who need to evaluate and choose the best practices for their work, goes well beyond the description or application of any single method. "Software is both a craft and a science, both a work of passion and a work of principle. Writing good software requires both wild flights of imagination and creativity, as well as the hard reality of engineering tradeoffs. This book is an attempt at describing that balance." —Robert Martin (unclebob) "The work of Ivar Jacobson and his colleagues, started as part of the SEMAT initiative, has taken a systematic approach to identifying a ‘kernel’ of software engineering principles and practices that have stood the test of time and recognition.” —Bertrand Meyer “The software development industry needs and demands a core kernel and language for defining software development practices—practices that can be mixed and matched, brought on board from other organizations; practices that can be measured; practices that can be integrated; and practices that can be compared and contrasted for speed, quality, and price. This thoughtful book gives a good grounding in ways to think about the problem, and a language to address the need, and every software engineer should read it.” —Richard Soley

As technology continues to evolve, the popularity of mobile computing has become inherent within today’s society. With the majority of the population using some form of mobile device, it has become increasingly important to develop more efficient cloud platforms. Modern Software Engineering Methodologies for Mobile and Cloud Environments investigates emergent trends and research on innovative software platforms in mobile and cloud computing. Featuring state-of-the-art software engineering methods, as well as new techniques being utilized in the field, this book is a pivotal reference source for professionals, researchers, practitioners, and students interested in mobile and cloud environments.

The first course in software engineering is the most critical. Education must start from an understanding of the heart of software development, from familiar ground that is common to all software development endeavors. This book is an in-depth introduction to software engineering that uses a systematic, universal kernel to teach the essential elements of all software engineering methods. This kernel, Essence, is a vocabulary for defining methods and practices. Essence was envisioned and originally created by Ivar Jacobson and his colleagues, developed by Software Engineering Method and Theory (SEMAT) and approved by The Object Management Group (OMG) as a standard in 2014. Essence is a practice-independent framework for thinking and reasoning about the practices we have and the practices we need. Essence establishes a shared and standard understanding of what is at the heart of software development. Essence is agnostic to any particular method, lifecycle independent, programming language independent, concise, scalable, extensible, and formally specified. Essence frees the practices from their method prisons. The first part of the book describes Essence, the essential elements to work with, the essential things to do and the essential competencies you need when developing software. The other three parts describe more and more advanced use cases of Essence. Using real but manageable examples, it covers the fundamentals of Essence and the innovative use of serious games to support software engineering. It also explains how current practices such as user stories, use cases, Scrum, and micro-services can be described using Essence, and illustrates how their activities can be represented using the Essence notions of cards and checklists. The fourth part of the book offers a vision how Essence can be scaled to support large, complex systems engineering. Essence is supported by an ecosystem developed and maintained by a community of experienced people worldwide. From this ecosystem, professors and students can select what they need and create their own way of working, thus learning how to create ONE way of working that matches the particular situation and needs.

Are there any constraints known that bear on the ability to perform Agile Management for Software Engineering work? How is the team addressing them? In a project to restructure Agile Management for Software Engineering outcomes, which stakeholders would you involve? How much are sponsors, customers, partners, stakeholders involved in Agile Management for Software Engineering? In other words, what are the risks, if Agile Management for Software Engineering does not deliver successfully? How does the organization define, manage, and improve its Agile Management for Software Engineering processes? What are the business goals Agile Management for Software Engineering is aiming to achieve? Defining, designing, creating, and implementing a process to solve a business challenge or meet a business objective is the most valuable role... In EVERY company, organization and department. Unless you are talking a one-time, single-use project within a business, there should be a process. Whether that process is managed and implemented by humans, AI, or a combination of the two, it needs to be designed by someone with a complex enough perspective to ask the right questions. Someone capable of asking the right questions and step back and say, 'What are we really trying to accomplish here? And is there a different way to look at it?' For more than twenty years, The Art of Service's Self-Assessments empower people who can do just that - whether their title is marketer, entrepreneur, manager, salesperson, consultant, business process manager, executive assistant, IT Manager, CxO etc... - they are the people who rule the future. They are people who watch the process as it happens, and ask the right questions to make the process work better. This book is for managers, advisors, consultants, specialists, professionals and anyone interested in Agile Management for Software Engineering assessment. All the tools you need to an in-depth Agile Management for Software Engineering Self-Assessment. Featuring 616 new and updated case-based questions, organized into seven core areas of process design, this Self-Assessment will help you identify areas in which Agile Management for Software Engineering improvements can be made. In using the questions you will be better able to: - diagnose Agile Management for Software Engineering projects, initiatives, organizations, businesses and processes using accepted diagnostic standards and practices - implement evidence-based best practice strategies aligned with overall goals - integrate recent advances in Agile Management for Software Engineering and process design strategies into practice according to best practice guidelines Using a Self-Assessment tool known as the Agile Management for Software Engineering Scorecard, you will develop a clear picture of which Agile Management for Software Engineering areas need attention. Included with your purchase of the book is the Agile Management for Software Engineering Self-Assessment downloadable resource, which contains all questions and Self-Assessment areas of this book in a ready to use Excel dashboard, including the self-assessment, graphic insights, and project planning automation - all with examples to get you started with the assessment right away. Access instructions can be found in the book. You are free to use the Self-Assessment contents in your presentations and materials for customers without asking us - we are here to help.

A one-semester college course in software engineering focusing on cloud computing, software as a service (SaaS), and Agile development using Extreme Programming (XP). This book is neither a step-by-step tutorial nor a reference book. Instead, our goal is to bring a diverse set of software engineering topics together into a single narrative, help readers understand the most important ideas through concrete examples and a learn-by-doing approach, and teach readers enough about each topic to get them started in the field. Courseware for doing the work in the book is available as a virtual machine image that can be downloaded or deployed in the cloud. A free MOOC (massively open online course) at saas-class.org follows the book's content and adds programming assignments and quizzes. See http: //saasbook.info for details.

In Team Topologies DevOps consultants Matthew Skelton and Manuel Pais share secrets of successful team patterns and interactions to help readers choose and evolve the right team patterns for their organization, making sure to keep the software healthy and optimize value streams. Team Topologies will help readers discover: • Team patterns used by successful organizations. • Common team patterns to avoid with modern software systems. • When and why to use different team patterns • How to evolve teams effectively. • How to split software and align to teams.

"If we don't drop our price, we will lose the deal." That's the desperate cry from salespeople as they try to win deals in competitive marketplaces. While the easy answer is to lower the price, the company sacrifices margin--oftentimes unnecessarily. To win deals at the prices you want,the strategy needed is differentiation. Most executives think marketing is the sole source of differentiation. But what about the sales function of the company? This commonly neglected differentiation opportunity provides a multitude of ways to stand out from the competition. This groundbreaking book teaches you how to develop those strategies. In Sales Differentiation, sales management strategist, Lee B. Salz presents nineteen easy-to-implement concepts to help salespeople win deals while protecting margins. These concepts apply to any salesperson in any industry and are based on the foundation that "how you sell, not just what you sell, differentiates you." The strategies are presented in easy-to-understand stories and can quickly be put into practice. Divided into two sections, the "what you sell" chapters help salespeople: Recognize that the expression "we are the best" causes differentiation to backfire. Avoid the introspective question that frustrates salespeople and ask the right question to fire them up. Understand what their true differentiators are and how to effectively position them with buyers. Find differentiators in every nook and cranny of the company using the six components of the "Sales Differentiation Universe." Create strategies to position differentiators so buyers see value in them. The "how you sell" section teaches salespeople how to provide meaningful value to buyers and differentiate themselves in every stage of the sales process. This section helps salespeople: Develop strategies to engage buyers and turn buyer objections into sales differentiation opportunities. Shape buyer decision criteria around differentiators. Turn a commoditized Request for Proposal (RFP) process into a differentiation opportunity. Use a buyer request for references as a way to stand out from the competition. Leverage the irrefutable, most powerful differentiator...themselves. Whether you've been selling for twenty years or are new to sales, the tools you learn in Sales Differentiation will help you knock-out the competition, build profitable new relationships, and win deals at the prices you want.

Solid requirements engineering has increasingly been recognized as the key to improved, on-time, and on-budget delivery of software and systems projects. This textbook provides a comprehensive treatment of the theoretical and practical aspects of discovering, analyzing, modeling, validating, testing, and writing requirements for systems of all kinds, with an intentional focus on software-intensive systems. It brings into play a variety of formal methods, social models, and modern requirements for writing techniques to be useful to the practicing engineer. This book was written to support both undergraduate and graduate requirements engineering courses. Each chapter includes simple, intermediate, and advanced exercises. Advanced exercises are suitable as a research assignment or independent study and are denoted by an asterisk. Various exemplar systems illustrate points throughout the book, and four systems in particular—a baggage handling system, a point of sale system, a smart home system, and a wet well pumping system—are used repeatedly. These systems involve application domains with which most readers are likely to be familiar, and they cover a wide range of applications from embedded to organic in both industrial and consumer implementations. Vignettes at the end of each chapter provide mini-case studies showing how the learning in the chapter can be employed in real systems. Requirements engineering is a dynamic field and this text keeps pace with these changes. Since the first edition of this text, there have been many changes and improvements. Feedback from instructors, students, and corporate users of the text was used to correct, expand, and improve the material. This third edition includes many new topics, expanded discussions, additional exercises, and more examples. A focus on safety critical systems, where appropriate in examples and exercises, has also been introduced. Discussions have also been added to address the important domain of the Internet of Things. Another significant change involved the transition from the retired IEEE Standard 830, which was referenced throughout previous editions of the text, to its successor, the ISO/IEC/IEEE 29148 standard.

A directory for up-and-coming jobs in the near-future employment market includes recommendations for finding or advancing a career and draws on statistics from the U.S. Department of Labor, in a guide that includes coverage of more than 250 occupations. Original.

Modern Software Sales EngineeringIt Isn't All Just Ping Pong and BeerCreatespace Independent Publishing Platform

A breakthrough approach to managing agile software development, Agile methods might just be the alternative to outsourcing. However, agile development must scale in scope and discipline to be acceptable in the boardrooms of the Fortune 1000. In Agile Management for Software Engineering, David J. Anderson shows managers how to apply management science to gain the full business benefits of agility through application of the focused approach taught by Eli Goldratt in his Theory of Constraints. Whether you're using XP, Scrum, FDD, or another agile approach, you'll learn how to develop management discipline for all phases of the engineering process, implement realistic financial and production metrics, and focus on building software that delivers maximum customer value and outstanding business results.Coverage includes: Making the business case for agile methods: practical tools and disciplines How to choose an agile method for your next project Breakthrough application of Critical Chain Project Management and constraint-driven control of the flow of value Defines the four new roles for the agile manager in software projects—and competitive IT organizations Whether you're a development manager, project manager, team leader, or senior IT executive, this book will help you achieve all four of your most urgent challenges: lower cost, faster delivery, improved quality, and focused alignment with the business.

John Care and Chris Daly lay out the 3+1 rules of SE Leadership. A simple framework designed for everyone - from SEs thinking about moving into management to the newest of new SE

Managers to a Global SE Vice President. This is a fascinating blend of tactical and strategic advice based on 30+ years of experience and many years of running SE specific workshops. All designed to allow you to follow the 3+1 Rules: Develop And Serve Your People, Run Pre-Sales As A Business, and Serve Your Customers all matched up with Rule #0 Manage Yourself. It's a common and often repeated story. You take a rock star Sales Engineer who is highly valued for their sales and business skills - and make them a manager because they are a great SE. With no regard for their possible leadership skills whatsoever. Perhaps they are pointed at a few online HR resources and take a mandatory "Managing Within The Law" session. Then they are released into the wild, and asked to manage, lead and motivate a team of Sales Engineers - each of whom performs the job differently than the newly minted manager used to do.

Every high-tech sales team today has technical pros on board to “explain how things work,” and this success-tested training resource is written just for them. This newly revised and expanded third edition of an Artech House bestseller offers invaluable insights and tips for every stage of the selling process. This third edition features a wealth of new material, including new chapters on business-driven discovery, white boarding, trusted advisors, and calculating ROI. This invaluable book equips new sales engineers with powerful sales and presentation techniques that capitalize on their technical background—all spelled out step-by-step by a pair of technical sales experts with decades of eye-popping, industry-giant success under their belt.

Writing and running software is now as much a part of science as telescopes and test tubes, but most researchers are never taught how to do either well. As a result, it takes them longer to accomplish simple tasks than it should, and it is harder for them to share their work with others than it needs to be. This book introduces the concepts, tools, and skills that researchers need to get more done in less time and with less pain. Based on the practical experiences of its authors, who collectively have spent several decades teaching software skills to scientists, it covers everything graduate-level researchers need to automate their workflows, collaborate with colleagues, ensure that their results are trustworthy, and publish what they have built so that others can build on it. The book assumes only a basic knowledge of Python as a starting point, and shows readers how it, the Unix shell, Git, Make, and related tools can give them more time to focus on the research they actually want to do. Research Software Engineering with Python can be used as the main text in a one-semester course or for self-guided study. A running example shows how to organize a small research project step by step; over a hundred exercises give readers a chance to practice these skills themselves, while a glossary defining over two hundred terms will help readers find their way through the terminology. All of the material can be re-used under a Creative Commons license, and all royalties from sales of the book will be donated to The Carpentries, an organization that teaches foundational coding and data science skills to researchers worldwide.

Minority women who have made it to the top offer tips and advice to others who wonder what it takes to succeed in careers in both the for-profit and nonprofit worlds.

A complete introduction to building robust and reliable software Beginning Software Engineering demystifies the software engineering methodologies and techniques that professional developers use to design and build robust, efficient, and consistently reliable software. Free of jargon and assuming no previous programming, development, or management experience, this accessible guide explains important concepts and techniques that can be applied to any programming language. Each chapter ends with exercises that let you test your understanding and help you elaborate on the chapter's main concepts. Everything you need to understand waterfall, Sashimi, agile, RAD, Scrum, Kanban, Extreme Programming, and many other development models is inside! Describes in plain English what software engineering is Explains the roles and responsibilities of team members working on a software engineering project Outlines key phases that any software engineering effort must handle to produce applications that are powerful and dependable Details the most popular software development methodologies and explains the different ways they handle critical development tasks Incorporates exercises that expand upon each chapter's main ideas Includes an extensive glossary of software engineering terms

Software Engineering for Science provides an in-depth collection of peer-reviewed chapters that describe experiences with applying software engineering practices to the development of scientific software. It provides a better understanding of how software engineering is and should be practiced, and which software engineering practices are effective for scientific software. The book starts with a detailed overview of the Scientific Software Lifecycle, and a general overview of the scientific software development process. It highlights key issues commonly arising during scientific software development, as well as solutions to these problems. The second part of the book provides examples of the use of testing in scientific software development, including key issues and challenges. The chapters then describe solutions and case studies aimed at applying testing to scientific software development efforts. The final part of the book provides examples of applying software engineering techniques to scientific software, including not only computational modeling, but also software for data management and analysis. The authors describe their experiences and lessons learned from developing complex scientific software in different domains. About the Editors Jeffrey Carver is an Associate Professor in the Department of Computer Science at the University of Alabama. He is one of the primary organizers of the workshop series on Software Engineering for Science (http://www.SE4Science.org/workshops). Neil P. Chue Hong is Director of the Software Sustainability Institute at the University of Edinburgh. His research interests include barriers and incentives in research software ecosystems and the role of software as a research object. George K. Thiruvathukal is Professor of Computer Science at Loyola University Chicago and Visiting Faculty at Argonne National Laboratory. His current research is focused on software metrics in open source mathematical and scientific software.

The overwhelming majority of a software system’s lifespan is spent in use, not in design or implementation. So, why does conventional wisdom insist that software engineers focus primarily on the design and development of large-scale computing systems? In this collection of essays and articles, key members of Google’s Site Reliability Team explain how and why their commitment to the entire lifecycle has enabled the company to successfully build, deploy, monitor, and maintain some of the largest software systems in the world. You’ll learn the principles and practices that enable Google engineers to make systems more scalable, reliable, and efficient—lessons directly applicable to your organization. This book is divided into four sections: Introduction—Learn what site reliability engineering is and why it differs from conventional IT industry practices Principles—Examine the patterns, behaviors, and areas of concern that influence the work of a site reliability engineer (SRE) Practices—Understand the theory and practice of an SRE’s day-to-day work: building and operating large distributed computing systems Management—Explore Google's best practices for training, communication, and meetings that your organization can use

Discover the untapped features of object-oriented programming and use it with other software tools to code fast, efficient applications. Key Features Explore the complexities of object-oriented programming (OOP) Discover what OOP can do for you Learn to use the key tools and software engineering practices to support your own programming needs Book Description Your experience and knowledge always influence the approach you take and the tools you use to write your programs. With a sound understanding of how to approach your goal and what software paradigms to use, you can create high-performing applications quickly and efficiently. In this two-part book, you’ll discover the untapped features of object-oriented programming and use it with other software tools to code fast and efficient applications. The first part of the book begins with a discussion on how OOP is used today and moves on to analyze the ideas and problems that OOP doesn’t address. It continues by deconstructing the complexity of OOP, showing you its fundamentally simple core. You’ll see that, by using the distinctive elements of OOP, you can learn to build your applications more easily. The next part of this book talks about acquiring the skills to become a better programmer. You’ll get an overview of how various tools, such as version control and build management, help make your life easier. This book also discusses the pros and cons of other programming paradigms, such as aspect-oriented programming and functional programming, and helps to select the correct approach for your projects. It ends by talking about the philosophy behind

designing software and what it means to be a "good" developer. By the end of this two-part book, you will have learned that OOP is not always complex, and you will know how you can evolve into a better programmer by learning about ethics, teamwork, and documentation. What you will learn Untangle the complexity of object-oriented programming by breaking it down to its essential building blocks Realize the full potential of OOP to design efficient, maintainable programs Utilize coding best practices, including TDD, pair programming and code reviews, to improve your work Use tools, such as source control and IDEs, to work more efficiently Learn how to most productively work with other developers Build your own software development philosophy Who this book is for This book is ideal for programmers who want to understand the philosophy behind creating software and what it means to be “good” at designing software. Programmers who want to deconstruct the OOP paradigm and see how it can be reconstructed in a clear, straightforward way will also find this book useful. To understand the ideas expressed in this book, you must be an experienced programmer who wants to evolve their practice.
Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world’s leading practitioners construct and maintain software. This book covers Google’s unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You’ll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

"This book presents current, effective software engineering methods for the design and development of modern Web-based applications"--Provided by publisher.

Software Engineering with Microsoft Visual Studio Team System is written for any software team that is considering running a software project using Visual Studio Team System (VSTS), or evaluating modern software development practices for its use. It is about the value-up paradigm of software development, which forms the basis of VSTS: its guiding ideas, why they are presented in certain ways, and how they fit into the process of managing the software lifecycle. This book is the next best thing to having an onsite coach who can lead the team through a consistent set of processes. Sam Guckenheimer has been the chief customer advocate for VSTS, responsible for its end-to-end external design. He has written this book as a framework for thinking about software projects in a way that can be directly tooled by VSTS. It presents essential theory and practical examples to describe a realistic process for IT projects. Readers will learn what they need to know to get started with VSTS, including The role of the value-up paradigm (versus work-down) in the software development lifecycle, and the meanings and importance of “flow” The use of MSF for Agile Software Development and MSF for CMMI Process Improvement Work items for planning and managing backlog in VSTS Multidimensional, daily metrics to maintain project flow and enable estimation Creating requirements using personas and scenarios Project management with iterations, trustworthy transparency, and friction-free metrics Architectural design using a value-up view, service-oriented architecture, constraints, and qualities of service Development with unit tests, code coverage, profiling, and build automation Testing for customer value with scenarios, qualities of service, configurations, data, exploration, and metrics Effective bug reporting and bug assessment Troubleshooting a project: recognizing and correcting common pitfalls and antipatterns This is a book that any team using or considering VSTS should read.

Software startups make global headlines every day. As technology companies succeed and grow, so do their engineering departments. In your career, you'll may suddenly get the opportunity to lead teams: to become a manager. But this is often uncharted territory. How can you decide whether this career move is right for you? And if you do, what do you need to learn to succeed? Where do you start? How do you know that you're doing it right? What does "it" even mean? And isn't management a dirty word? This book will share the secrets you need to know to manage engineers successfully. Going from engineer to manager doesn't have to be intimidating. Engineers can be managers, and fantastic ones at that. Cast aside the rhetoric and focus on practical, hands-on techniques and tools. You'll become an effective and supportive team leader that your staff will look up to. Start with your transition to being a manager and see how that compares to being an engineer. Learn how to better organize information, feel productive, and delegate, but not micromanage. Discover how to manage your own boss, hire and fire, do performance and salary reviews, and build a great team. You'll also learn the psychology: how to ship while keeping staff happy, coach and mentor, deal with deadline pressure, handle sensitive information, and navigate workplace politics. Consider your whole department. How can you work with other teams to ensure best practice? How do you help form guilds and committees and communicate effectively? How can you create career tracks for individual contributors and managers? How can you support flexible and remote working? How can you improve diversity in the industry through your own actions? This book will show you how. Great managers can make the world a better place. Join us.

True or false? In selling high-value products or services: 'closing' increases your chance of success; it is essential to describe the benefits of your product or service to the customer; objection handling is an important skill; open questions are more effective than closed questions. All false, says this provocative book. Neil Rackham and his team studied more than 35,000 sales calls made by 10,000 sales people in 23 countries over 12 years. Their findings revealed that many of the methods developed for selling low-value goods just don‘t work for major sales. Rackham went on to introduce his SPIN-Selling method. SPIN describes the whole selling process: Situation questions Problem questions Implication questions Need-payoff questions SPIN-Selling provides you with a set of simple and practical techniques which have been tried in many of today‘s leading companies with dramatic improvements to their sales performance.

This is the eagerly-anticipated revision to one of the seminal books in the field of software architecture which clearly defines and explains the topic.

Based on the popular Artech House classic, Digital Communication Systems Engineering with Software-Defined Radio, this book provides a practical approach to quickly learning the software-defined radio (SDR) concepts needed for work in the field. This up-to-date volume guides readers on how to quickly prototype wireless designs using SDR for real-world testing and experimentation. This book explores advanced wireless communication techniques such as OFDM, LTE, WLA, and hardware targeting. Readers will gain an understanding of the core concepts behind wireless hardware, such as the radio frequency front-end, analog-to-digital and digital-to-analog converters, as well as various processing technologies. Moreover, this volume includes chapters on timing estimation, matched filtering, frame synchronization message decoding, and source coding. The orthogonal frequency division multiplexing is explained and details about HDL code generation and deployment are provided. The book concludes with coverage of the WLAN toolbox with OFDM beacon reception and the LTE toolbox with downlink reception.

Multiple case studies are provided throughout the book. Both MATLAB and Simulink source code are included to assist readers with their projects in the field.
This Expert Guide gives you the techniques and technologies in software engineering to optimally design and implement your embedded system. Written by experts with a solutions focus, this encyclopedic reference gives you an indispensable aid to tackling the day-to-day problems when using software engineering methods to develop your embedded systems. With this book you will learn: The principles of good architecture for an embedded system Design practices to help make your embedded project successful Details on principles that are often a part of embedded systems, including digital signal processing, safety-critical principles, and development processes Techniques for setting up a performance engineering strategy for your embedded system software How to develop user interfaces for embedded systems Strategies for testing and deploying your embedded system, and ensuring quality development processes Practical techniques for optimizing embedded software for performance, memory, and power Advanced guidelines for developing multicore software for embedded systems How to develop embedded software for networking, storage, and automotive segments How to manage the embedded development process Includes contributions from: Frank Schirrmeister, Shelly Gretlein, Bruce Douglass, Erich Styger, Gary Stringham, Jean Labrosse, Jim Trudeau, Mike Brogioli, Mark Pitchford, Catalin Dan Udma, Markus Levy, Pete Wilson, Whit Waldo, Inga Harris, Xinxin Yang, Srinivasa Addepalli, Andrew McKay, Mark Kraeling and Robert Oshana. Road map of key problems/issues and references to their solution in the text Review of core methods in the context of how to apply them Examples demonstrating timeless implementation details Short and to- the- point case studies show how key ideas can be implemented, the rationale for choices made, and design guidelines and trade-offs
The idea of editing a book on modern software architectures and tools for CAPE (Computer Aided Process Engineering) came about when the editors of this volume realized that existing titles relating to CAPE did not include references to the design and development of CAPE software. Scientific software is needed to solve CAPE related problems by industry/academia for research and development, for education and training and much more. There are increasing demands for CAPE software to be versatile, flexible, efficient, and reliable. This means that the role of software architecture is also gaining increasing importance. Software architecture needs to reconcile the objectives of the software; the framework defined by the CAPE methods; the computational algorithms; and the user needs and tools (other software) that help to develop the CAPE software. The object of this book is to bring to the reader, the software side of the story with respect to computer aided process engineering.