

Software Engineering Global Edition

This comprehensive approach to the creation of software systems charts a road through system modelling techniques, allowing software engineers to create software meeting two very basic requirements: • that the software system represent a narrow emulation of the organization system that served as its model; • and that the software system display life attributes identical to those of the organization system that it automatizes. The result is a quantum leap increase in software application quality. Such benefit is achieved by the introduction of a fundamental paradigm: the office-floor metaphor which incorporates such well-balanced basic ideas as the functional normalization of tasks and information (in sharp contrast to the classic data normalization) and the principle of tenant-ownership.

"This book provides analysis, characterization and refinement of software engineering data in terms of machine learning methods. It depicts applications of several machine learning approaches in software systems development and deployment, and the use of machine learning methods to establish predictive models for software quality while offering readers suggestions by proposing future work in this emerging research field"--Provided by publisher.

This custom edition is published for the University of Southern Queensland.

Managing Humans is a selection of the best essays from Michael Lopp's popular website Rands in Repose (www.randsinrepose.com). Lopp is one of the most sought-after IT managers in Silicon Valley, and draws on his experiences at Apple, Netscape, Symantec, and Borland. This book reveals a variety of different approaches for creating innovative, happy development teams. It covers handling conflict, managing wildly differing personality types, infusing innovation into insane product schedules, and figuring out how to build lasting and useful engineering culture. The essays are biting, hilarious, and always informative.

Research Software Engineering with Python

Software Engineering, Global Edition

Global Software Engineering

Virtualization and Coordination

Software Engineering with Objects and Components

Lessons Learned from Programming Over Time

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

To provide the necessary security and quality assurance activities into Internet of Things (IoT)-based software development, innovative engineering practices are vital. They must be given an even higher level of importance than most other events in the field. Integrating the Internet of Things Into Software Engineering Practices provides research on the integration of IoT into the software development life cycle (SDLC) in terms of requirements management, analysis, design, coding, and testing, and provides security and quality assurance activities to IoT-based software development. The content within this publication covers agile software, language specification, and collaborative software and is designed for analysts, security experts, IoT software programmers, computer and software engineers, students, professionals, and researchers.

The XP conference series established in 2000 was the first conference dedicated to agile processes in software engineering. The idea of the conference is to offer a unique setting for advancing the state of the art in the research and practice of agile processes. This year's conference was the ninth consecutive edition of this international event. The conference has grown to be the largest conference on agile software development outside North America. The XP conference enjoys being one of those conferences that truly brings practitioners and academics together. About 70% of XP participants come from industry and the number of academics has grown steadily over the years. XP is more of an experience rather than a regular conference. It offers several different ways to interact and strives to create a truly collaborative environment where new ideas and exciting findings can be presented and shared. For example, this year's open space session, which was "a conference within a conference", was larger than ever before. Agile software development is a unique phenomenon from several perspectives.

This is the eBook of the printed book and may not include any media, website access codes, or print supplements that may be packaged with the bound book. Intended for introductory and advanced courses in software engineering. The ninth edition of Software Engineering presents a broad perspective of software engineering, focusing on the processes and techniques fundamental to the creation of reliable, software systems. Increased coverage of agile methods and software reuse, along with coverage of 'traditional' plan-driven software engineering, gives readers the most up-to-date view of the field currently available. Practical case studies, a full set of easy-to-access supplements, and extensive web resources make teaching the course easier than ever. The book is now structured into four parts: 1: Introduction to Software Engineering 2: Dependability and Security 3: Advanced Software Engineering 4: Software Engineering Management
Advancements in Model-Driven Architecture in Software Engineering
Integrating the Internet of Things Into Software Engineering Practices
Methods, Practical Techniques, and Applications

A Guide to Data Capture, Content, and Transactions
 Designing, Engineering, and Analyzing Reliable and Efficient Software
 How to Be the Leader Your Development Team Needs

Computer science graduates often find software engineering knowledge and skills are more in demand after they join the industry. However, given the lecture-based curriculum present in academia, it is not an easy undertaking to deliver industry-standard knowledge and skills in a software engineering classroom as such lectures hardly engage or convince students. *Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills* combines recent advances and best practices to improve the curriculum of software engineering education. This book is an essential reference source for researchers and educators seeking to bridge the gap between industry expectations and what academia can provide in software engineering education.

Users increasingly demand more from their software than ever before—more features, fewer errors, faster runtimes. To deliver the best quality products possible, software engineers are constantly in the process of employing novel tools in developing the latest software applications. *Progressions and Innovations in Model-Driven Software Engineering* investigates the most recent and relevant research on model-driven engineering. Within its pages, researchers and professionals in the field of software development, as well as academics and students of computer science, will find an up-to-date discussion of scientific literature on the topic, identifying opportunities and advantages, and complexities and challenges, inherent in the future of software engineering.

Perspectives on Data Science for Software Engineering presents the best practices of seasoned data miners in software engineering. The idea for this book was created during the 2014 conference at Dagstuhl, an invitation-only gathering of leading computer scientists who meet to identify and discuss cutting-edge informatics topics. At the 2014 conference, the concept of how to transfer the knowledge of experts from seasoned software engineers and data scientists to newcomers in the field highlighted many discussions. While there are many books covering data mining and software engineering basics, they present only the fundamentals and lack the perspective that comes from real-world experience. This book offers unique insights into the wisdom of the community's leaders gathered to share hard-won lessons from the trenches. Ideas are presented in digestible chapters designed to be applicable across many domains. Topics included cover data collection, data sharing, data mining, and how to utilize these techniques in successful software projects. Newcomers to software engineering data science will learn the tips and tricks of the trade, while more experienced data scientists will benefit from war stories that show what traps to avoid. Presents the wisdom of community experts, derived from a summit on software analytics Provides contributed chapters that share discrete ideas and technique from the trenches Covers top areas of concern, including mining security and social data, data visualization, and cloud-based data Presented in clear chapters designed to be applicable across many domains

Like other sciences and engineering disciplines, software engineering requires a cycle of model building, experimentation, and learning. Experiments are valuable tools for all software engineers who are involved in evaluating and choosing between different methods, techniques, languages and tools. The purpose of *Experimentation in Software Engineering* is to introduce students, teachers, researchers, and practitioners to empirical studies in software engineering, using controlled experiments. The introduction to experimentation is provided through a process perspective, and the focus is on the steps that we have to go through to perform an experiment. The book is divided into three parts. The first part provides a background of theories and methods used in experimentation. Part II then devotes one chapter to each of the five experiment steps: scoping, planning, execution, analysis, and result presentation. Part III completes the presentation with two examples. Assignments and statistical material are provided in appendixes. Overall the book provides indispensable information regarding empirical studies in particular for experiments, but also for case studies, systematic literature reviews, and surveys. It is a revision of the authors' book, which was published in 2000. In addition, substantial new material, e.g. concerning systematic literature reviews and case study research, is introduced. The book is self-contained and it is suitable as a course book in undergraduate or graduate studies where the need for empirical studies in software engineering is stressed. Exercises and assignments are included to combine the more theoretical material with practical aspects. Researchers will also benefit from the book, learning more about how to conduct empirical studies, and likewise practitioners may use it as a "cookbook" when evaluating new methods or techniques before implementing them in their organization.

Guide to Advanced Empirical Software Engineering

Software Engineering for Internet Applications

Progressions and Innovations in Model-Driven Software Engineering

Introduction to Software Engineering (Custom Edition)

Perspectives on Data Science for Software Engineering

Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications

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.

Computer Science: An Overview uses broad coverage and clear exposition to present a complete picture of the dynamic computer science field. Accessible to students from all backgrounds, Glenn Brookshear uses a language-independent context to encourage the development of a practical, realistic understanding of the field. An overview of each of the important areas of Computer Science (e.g. Networking, OS, Computer Architecture, Algorithms) provides students with a general level of proficiency for future courses. The Eleventh Edition features two new contributing authors (David Smith — Indiana University of PA; Dennis Brylow — Marquette University), new, modern examples, and updated coverage based on current technology.

The highly dynamic world of information technology service management stresses the benefits of the

quick and correct implementation of IT services. A disciplined approach relies on a separate set of assumptions and principles as an agile approach, both of which have complicated implementation processes as well as copious benefits. Combining these two approaches to enhance the effectiveness of each, while difficult, can yield exceptional dividends. Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products is an essential publication that focuses on clarifying theoretical foundations of balanced design methods with conceptual frameworks and empirical cases. Highlighting a broad range of topics including business trends, IT service, and software development, this book is ideally designed for software engineers, software developers, programmers, information technology professionals, researchers, academicians, and students.

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.

Software Engineering

Balancing Agile and Disciplined Engineering and Management Approaches for IT Services and Software Products

9th International Conference, XP 2008, Limerick, Ireland, June 10-14, 2008, Proceedings

Biting and Humorous Tales of a Software Engineering Manager

Engineering Software Products

Beginning Software Engineering

More software engineers are likely to work in a globally distributed environment, which brings benefits that include quick and better software development, less manpower retention, scalability, and less software development cost and sharing of knowledge from the global pool of employees. However, these work environments also introduce a physical separation between team members and project leaders, which can create problems in communication and ultimately lead to the failure of the project. Human Factors in Global Software Engineering is a collection of innovative research focusing on the challenges, issues, and importance of human factors in global software engineering organizations in order to help these organizations better manage their manpower and provide an appropriate culture and technology in order to make their software development projects successful. While highlighting topics including agile software, knowledge management, and human-computer interaction, this book is ideally designed for project managers, administrators, business professionals, researchers, practitioners, students, and academicians.

Technology and organizations co-evolve, as is illustrated by the growth of information and communication technology (ICT) and global software engineering (GSE). Technology has enabled the development of innovations in GSE. The literature on GSE has emphasized the role of the organization at the expense of technology. This book explores the role of technology in the evolution of globally distributed software engineering. To date, the role of the organization has been examined in coordinating GSE activities because of the prevalence of the logic of rationality (i.e., the efficiency ethos, mechanical methods, and mathematical analysis) and indeterminacy (i.e., the effectiveness ethos, natural methods, and functional analysis). This logic neglects the coordination role of ICT. However, GSE itself is an organizational mode that is technology-begotten, technology-dominated, and technology-driven, as is its coordination. GSE is a direct reflection of ICT innovation, change, and use, yet research into the role technology of GSE has been neglected. Global Software Engineering: Virtualization and Coordination considers existing fragmented explanations and perspectives in GSE research, poses new questions about GSE, and proposes a framework based on the logic of virtuality (i.e., creativity ethos, electrical methods, and technological analysis) rather than of rationality and indeterminacy. Virtuality is the primary perspective in this book's comprehensive study of GSE. The book concludes with an integrated explanation of GSE coordination made possible through ICT connectivity and capitalization.

Due to the role of software systems in safety-critical applications and in the satisfaction of customers and organizations, the development of efficient software engineering is essential. Designing, Engineering, and Analyzing Reliable and Efficient Software discusses and analyzes various designs, systems, and advancements in software engineering. With its coverage on the integration of mathematics, computer science, and practices in engineering, this book highlights the importance of ensuring and maintaining reliable software and is an essential resource for practitioners, professors and students in these fields of study.

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

Become an Effective Software Engineering Manager

Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills

Using UML

Computer Science

Foundations of Software Engineering

Software Engineering: Effective Teaching and Learning Approaches and Practices

This book discusses a comprehensive spectrum of software engineering techniques and shows how they can be applied in practical software projects. This edition features updated chapters on critical systems, project management and software requirements.

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.

This book, written from a software engineering point of view, provides the practitioner's guide to developing global e-commerce sites.

The best way to learn software engineering is by understanding its core and peripheral areas. Foundations of Software Engineering provides in-depth coverage of the areas of software engineering that are essential for becoming proficient in the field. The book devotes a complete chapter to each of the core areas. Several peripheral areas are also explained by assigning a separate chapter to each of them. Rather than using UML or other formal notations, the content in this book is explained in easy-to-understand language. Basic programming knowledge using an object-oriented language is helpful to understand the material in this book. The knowledge gained from this book can be readily used in other relevant courses or in real-world software development environments. This textbook educates students in software engineering principles. It covers almost all facets of software engineering, including requirement engineering, system specifications, system modeling, system architecture, system implementation, and system testing. Emphasizing practical issues, such as feasibility studies, this book explains how to add and develop software requirements to evolve software systems. This book was written after receiving feedback from several professors and software engineers. What resulted is a textbook on software engineering that not only covers the theory of software engineering but also presents real-world insights to aid students in proper implementation. Students learn key concepts through carefully explained and illustrated theories, as well as concrete examples and a complete case study using

Java. Source code is also available on the book's website. The examples and case studies increase in complexity as the book progresses to help students build a practical understanding of the required theories and applications.

An Introduction to Modern Software Engineering

Introduction to Software Engineering

Effective Teaching and Learning Approaches and Practices

Engineering Global E-Commerce Sites

Agile Processes in Software Engineering and Extreme Programming

Software Engineering for Embedded Systems

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

Software engineering lies at the heart of the computer revolution. Software is used in automobiles, airplanes, and many home appliances. As the boundaries between the telecommunications, entertainment, and computer industries continue to blur in multimedia and networking, the need for software will only increase, and software will become increasingly complex. Introduction to Software Engineering gives your students the fundamentals of this growing and rapidly changing field. The book highlights the goals of software engineering, namely to write programs that have all the following attributes: efficient, reliable, usable, modifiable, portable, testable, reusable, maintainable, compatible and correct. The nine chapters cover topics that include project management, defining requirements, software design, coding, testing and integration, delivery and installation, documentation, maintenance, and research issues. The author uses a hybrid approach, combining object-oriented technology and classical programming techniques to solve computing problems. He also places a strong emphasis on Internet technology and resources. A simple, but non-trivial, running example illustrates all stages of the software engineering process. In addition, where applicable, he covers the impact of Internet technology. Introduction to Software Engineering presents the basics of software engineering in a concise and direct format. With emphasis on Internet technology, software tools for programming, and hands-on learning, this book effectively prepares students to move from an educational situation towards applying their knowledge to the complex projects faced in the professional arena. Features

As the software industry continues to evolve, professionals are continually searching for practices that can assist with the various problems and challenges in information technology (IT). Agile development has become a popular method of research in recent years due to its focus on adapting to change. There are many factors that play into this process, so success is no guarantee. However, combining agile development with other software engineering practices could lead to a high rate of success in problems that arise during the maintenance and development of computing technologies. Software Engineering for Agile Application Development is a collection of innovative research on the methods and implementation of adaptation practices in software development that improve the quality and performance of IT products. The presented materials combine theories from current empirical research results as well as practical experiences from real projects that provide insights into incorporating agile qualities into the architecture of the software so that the product adapts to changes and is easy to maintain. While highlighting topics including continuous integration, configuration management, and business modeling, this book is ideally designed for software engineers, software developers, engineers, project managers, IT specialists, data scientists, computer science professionals, researchers, students, and academics.

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

Implications of Globalization

An Overview

Software Engineering at Google

Advances in Machine Learning Applications in Software Engineering

Human Factors in Global Software Engineering

Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization

An integral element of software engineering is model engineering. They both endeavor to minimize cost, time, and risks with quality software. As such, model engineering is a highly useful field that demands in-depth research on the most current approaches and techniques. Only by understanding the most up-to-date research can these methods reach their fullest potential. Advancements in Model-Driven Architecture in Software Engineering is an essential publication that prepares readers to exercise modeling and model transformation and covers state-of-the-art research and developments on various approaches for methodologies and platforms of model-driven architecture, applications and software development of model-driven architecture, modeling languages, and modeling tools. Highlighting a broad range of topics including cloud computing, service-oriented architectures, and modeling languages, this book is ideally designed for engineers, programmers, software designers, entrepreneurs, researchers, academicians, and students.

Economics and technology have dramatically re-shaped the landscape of software development. It is no longer uncommon to find a software development team dispersed across countries or continents. Geographically distributed development challenges the ability to clearly communicate, enforce standards, ensure quality levels, and coordinate tasks. Global Software Development Handbook explores techniques that can bridge distances, create cohesion, promote quality, and strengthen lines of communication. The book introduces techniques proven successful at international electronics and software giant Siemens AG. It shows how this multinational uses a high-level process framework that balances agility and discipline for globally distributed software development. The authors delineate an organizational structure that not only fosters team building, but also achieves effective collaboration among the central and satellite teams. The handbook explores the issues surrounding quality and the

processes required to realize quality in a distributed environment. Communication is a tremendous challenge, especially for teams separated by several time zones, and the authors elucidate how to uncover patterns of communication among these teams to determine effective strategies for managing communication. The authors analyze successful and failed projects and apply this information to how a project can be successful with distributed teams. They also provide lightweight processes that can be dynamically adapted to the demands of any project.

Do you... Use a computer to perform analysis or simulations in your daily work? Write short scripts or record macros to perform repetitive tasks? Need to integrate off-the-shelf software into your systems or require multiple applications to work together? Find yourself spending too much time working the kinks out of your code? Work with software engineers on a regular basis but have difficulty communicating or collaborating? If any of these sound familiar, then you may need a quick primer in the principles of software engineering. Nearly every engineer, regardless of field, will need to develop some form of software during their career. Without exposure to the challenges, processes, and limitations of software engineering, developing software can be a burdensome and inefficient chore. In *What Every Engineer Should Know about Software Engineering*, Phillip Laplante introduces the profession of software engineering along with a practical approach to understanding, designing, and building sound software based on solid principles. Using a unique question-and-answer format, this book addresses the issues and misperceptions that engineers need to understand in order to successfully work with software engineers, develop specifications for quality software, and learn the basics of the most common programming languages, development approaches, and paradigms.

After completing this self-contained course on server-based Internet applications software that grew out of an MIT course, students who start with only the knowledge of how to write and debug a computer program will have learned how to build sophisticated Web-based applications.

Software Engineering for Modern Web Applications: Methodologies and Technologies

Concepts, Methodologies, Tools, and Applications

Software Engineering for Agile Application Development

Building software that makes research possible

Global Software Development Handbook

For courses in computer science and software engineering *The Fundamental Practice of Software Engineering* Software Engineering introduces students to the overwhelmingly important subject of software programming and development. In the past few years, computer systems have come to dominate not just our technological growth, but the foundations of our world's major industries. This text seeks to lay out the fundamental concepts of this huge and continually growing subject area in a clear and comprehensive manner. The Tenth Edition contains new information that highlights various technological updates of recent years, providing students with highly relevant and current information. Sommerville's experience in system dependability and systems engineering guides the text through a traditional plan-based approach that incorporates some novel agile methods. The text strives to teach the innovators of tomorrow how to create software that will make our world a better, safer, and more advanced place to live.

Software Engineering, Global Edition

"This book provides integrated chapters on software engineering and enterprise systems focusing on parts integrating requirements engineering, software engineering, process and frameworks, productivity technologies, and enterprise systems"--Provided by publisher.

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

Managing Humans

Experimentation in Software Engineering

Delivering Non-Technical Knowledge and Skills

What Every Engineer Should Know about Software Engineering

Methodologies and Technologies

A Discipline of Software Engineering

Over the past decade, software engineering has developed into a highly respected field. Though computing and software engineering education continues to emerge as a prominent interest area of study, few books specifically focus on software engineering education itself. Software Engineering: Effective Teaching and Learning Approaches and Practices presents the latest developments in software engineering education, drawing contributions from over 20 software engineering educators from around the globe. Encompassing areas such as student assessment and learning, innovative teaching methods, and educational technology, this much-needed book greatly enhances libraries with its unique research content.