

Software Engineering Pfleeger 4th Edition

The book describes how to manage and successfully deliver large, complex, and expensive systems that can be composed of millions of line of software code, being developed by numerous groups throughout the globe, that interface with many hardware items being developed by geographically dispersed companies, where the system also includes people, policies, constraints, regulations, and a myriad of other factors. It focuses on how to seamlessly integrate systems, satisfy the customer's requirements, and deliver within the budget and on time. The guide is essentially a "shopping list" of all the activities that could be conducted with tailoring guidelines to meet the needs of each project.

This book explores the domain of software maintenance management and provides road maps for improving software maintenance organizations. It describes full maintenance maturity models organized by levels 1, 2, and 3, which allow for benchmarking and continuous improvement paths. Goals for each key practice area are also provided, and the model presented is fully aligned with the architecture and framework of software development maturity models of CMMI and ISO 15504. It is complete with case studies, figures, tables, and graphs. This book is a broad discussion covering the entire software development lifecycle. It uses a comprehensive case study to address each topic and features the following: A description of the development, by the fictional company Homeowner, of the DigitalHome (DH) System, a system with "smari" devices for controlling home lighting, temperature, humidity, small appliance power, and security A set of scenarios that provide a realistic framework for use of the DH System material Just-in-time training: each chapter includes mini tutorials introducing various software engineering topics that are discussed in that chapter and used in the case study A set of case study exercises that provide an opportunity to engage students in software development practice, either individually or in a team environment. Offering a new approach to learning about software engineering theory and practice, the text is specifically designed to: Support teaching software engineering, using a comprehensive case study covering the complete software development lifecycle Offer opportunities for students to actively learn about and engage in software engineering practice Provide a realistic environment to study a wide array of software engineering topics including agile development Software Engineering Practice: A Case Study Approach supports a student-centered, "active" learning style of teaching. The DH case study exercises provide a variety of opportunities for students to engage in realistic activities related to the theory and practice of software engineering. The text uses a fictitious team of software engineers to portray the nature of software engineering and to depict what actual engineers do when practicing software engineering. All the DH case study exercises can be used as team or group exercises in collaborative learning. Many of the exercises have specific goals related to team building and teaming skills. The text also can be used to support the professional development or certification of practicing software engineers. The case study exercises can be integrated with presentations in a workshop or short course for professionals.

The authors have designed a tutorial text to provide scientists with a technical understanding of computer-based imaging systems and how these systems interact with digital image processing algorithms. Contents include Boolean logic, image processing, image compression, basic computer architecture, advanced architectures, image processors, operating systems, error detection and correction, local area networks, object-oriented design paradigms, and software engineering. Contains numerous figures and case studies. Annotation copyrighted by Book News, Inc., Portland, OR

Security in Computing

Introduction to Computer-based Imaging Systems

A Case Study Approach

9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, Proceedings

A Threat/vulnerability/countermeasure Approach

Hearing Before the Subcommittee on Emerging Threats and Capabilities of the Committee on Armed Services, House of Representatives, One Hundred Twelfth Congress, First Session, Hearing Held February 11, 2011

Vols. 8-10 of the 1965-1984 master cumulation constitute a title index.

This book constitutes the refereed proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (formerly UML conferences), MoDELS 2006. The book presents 51 revised full papers and 2 invited papers. Discussion is organized in topical sections on evaluating UML, MDA in software development, concrete syntax, applying UML to interaction and coordination, aspects, model integration, formal semantics of UML, security, model transformation tools and implementation, and more.

Volume 54 presents six chapters on the changing face of software engineering-the process by which we build reliable software systems. We are constantly building faster and less expensive processors, which allow us to use different processes to try and conquer the "bug" problem facing all developments-how to build reliable systems with few errors at low or at least manageable cost. The first three chapters of this volume emphasize components and the impact that object-oriented design is having on the program development process (a current "hot topic"). The final three chapters present additional aspects of the software development process, including maintenance, purchasing strategies, and secure outsourcing of scientific computations.

Explores and identifies the main issues, concepts, principles and evolution of software testing, including software quality engineering and testing concepts, test data generation, test deployment analysis, and software test management This book examines the principles, concepts, and processes that are fundamental to the software testing function. This book is divided into five broad parts. Part I introduces software testing in the broader context of software engineering and explores the qualities that testing aims to achieve or ascertain, as well as the lifecycle of software testing. Part II covers mathematical foundations of software testing, which include software specification, program correctness and verification, concepts of software dependability, and a software testing taxonomy. Part III discusses test data generation, specifically, functional criteria and structural criteria. Test oracle design, test driver design, and test outcome analysis is covered in Part IV. Finally, Part V surveys managerial aspects of software testing, including software metrics, software testing tools, and software product line testing. Presents software testing, not as an isolated technique, but as part of an integrated discipline of software verification and validation Proposes program testing and program correctness verification within the same mathematical model, making it possible to deploy the two techniques in concert, by virtue of the law of diminishing returns Defines the concept of a software fault, and the related concept of relative correctness, and shows how relative correctness can be used to characterize monotonic fault removal Presents the activity of software testing as a goal oriented activity, and explores how the conduct of the test depends on the selected goal Covers all phases of the software testing lifecycle, including test data generation, test oracle design, test driver design, and test outcome analysis Software Testing: Concepts and Operations is a great resource for software quality and software engineering students because it presents them with fundamentals that help them to prepare for their ever evolving discipline.

International Symposium, MIS 2004, Salzburg, Austria, September 15-18, 2004, Revised Selected Papers

Evaluation and Continuous Improvement

Software Engineering

A Master Cumulation

A Student Guide to Object-Oriented Development

Trends in Software Engineering

This book constitutes the thoroughly refereed post-proceedings of the Metainformatics Symposium, MIS 2004, held in Salzburg, Austria in September 2004. The 17 revised full papers presented were carefully reviewed and selected for inclusion in the book. The papers are devoted to finding useful abstractions, notations, analytical frameworks, formalisms, and systems that improve the understanding of the underlying structure of various disciplines and families of systems within computer science.

It is my belief that software engineers not only need to know software engineering methods and processes, but that they also should know how to assess them. Conse quently, I have taught principles of experimentation and empirical studies as part of the software engineering curriculum. Until now, this meant selecting a text from another discipline, usually psychology, and augmenting it with journal or confer ence papers that provide students with software engineering examples of experi ments and empirical studies. This book fills an important gap in the software engineering literature: it pro vides a concise, comprehensive look at an important aspect of software engineer ing: experimental analysis of how well software engineering methods, methodologies, and processes work. Since all of these change so rapidly in our field, it is important to know how to evaluate new ones. This book teaches how to go about doing this and thus is valuable not only for the software engineering stu dent, but also for the practicing software engineering professional who will be able to • Evaluate software engineering techniques. • Determine the value (or lack thereof) of claims made about a software engineer ing method or process in published studies. Finally, this book serves as a valuable resource for the software engineering researcher.

Software EngineeringTheory and PracticePrentice Hall

Software configuration management (SCM) is one of the scientific tools that is aimed to bring control to the software development process. This new resource is a complete guide to implementing, operating, and maintaining a successful SCM system for software development.

Project managers, system designers, and software developers are presented with not only the basics of SCM, but also the different phases in the software development lifecycle and how SCM plays a role in each phase. The factors that should be considered and the pitfalls that should be avoided while designing the SCM system and SCM plan are also discussed. In addition, this third edition is updated to include cloud computing and on-demand systems. This book does not rely on one specific tool or standard for explaining the SCM concepts and techniques; In fact, it gives readers enough information about SCM, the mechanics of SCM, and SCM implementation, so that they can successfully implement a SCM system.

Software Testing and Quality Assurance

Encyclopedia of Information Science and Technology, Fourth Edition

Software Engineering Practice

A Methodical Approach, 2nd Edition

An Introduction

Book Review Index

Are you attracted by the promises of agile methods but put off by the fanaticism of many agile texts? Would you like to know which agile techniques work, which ones do not matter much, and which ones will harm your projects? Then you need Agile! the first exhaustive, objective and tools. Agile methods are one of the most important developments in software over the past decades, but also a surprising mix of the best and the worst. Until now every project and developer had to sort out the good ideas from the bad by themselves. This book spares you the presentation of agile techniques and a perceptive analysis of their benefits and limitations. Agile! serves first as a primer on agile development: one chapter each introduces agile principles, roles, managerial practices, technical practices and artifacts. A separate chapter analyzes Programming, Lean Software, Scrum and Crystal. The accompanying critical analysis explains what you should retain and discard from agile ideas. It is based on Meyer's thorough understanding of software engineering, and his extensive personal experience of programming and p limitations of agile methods as well as their truly brilliant contributions — even those to which their own authors do not do full justice. Three important chapters precede the core discussion of agile ideas: an overview, serving as a concentrate of the entire book; a dissection of and a review of classical software engineering techniques, such as requirements analysis and lifecycle models, which agile methods criticize. The final chapters describe the precautions that a company should take during a transition to agile development and present an overall a book to discuss agile methods, beyond the brouhaha, in the general context of modern software engineering. It is a key resource for projects that want to combine the best of established results and agile innovations.

Each and every chapter covers the contents up to a reasonable depth necessary for the intended readers in the field. The book consists in all about 1200 exercises based on the topics and sub-topics covered. Keeping in view the emerging trends in newly emerging scenario with book specially includes the following chapters, but not limited to these only. This book explains all the notions related to software engineering in a very systematic way, which is of utmost importance to the novice readers in the field of software Engineering.

Modern information systems differ in essence from their predecessors. They support operations at multiple locations and different time zones, are distributed and network-based, and use multidimensional data analysis, data warehousing, knowledge discovery, knowledge management information processing methods. This book considers fundamental issues of modern information systems. It discusses query processing, data quality, data mining, knowledge management, mobile computing, software engineering for information systems construction, and other t that are not available elsewhere. With more than 40 contributors, it is a solid source of information about the state of the art in the field of databases and information systems. It is intended for researchers, advanced students, and practitioners who are concerned with the de "If the purpose is to create one of the best books on requirements yet written, the authors have succeeded." —Capers Jones Software can solve almost any problem. The trick is knowing what the problem is. With about half of all software errors originating in the requirements of the problem is needed. Getting the requirements right is crucial if we are to build systems that best meet our needs. We know, beyond doubt, that the right requirements produce an end result that is as innovative and beneficial as it can be, and that system development is Requirements Process: Getting Requirements Right, Third Edition, sets out an industry-proven process for gathering and verifying requirements, regardless of whether you work in a traditional or agile development environment. In this sweeping update of the bestselling guide, th what the customer wants and needs, in the most efficient manner possible. Features include The Volere requirements process for discovering requirements, for use with both traditional and iterative environments A specification template that can be used as the basis for your o guides that help you funnel your efforts into only the requirements work needed for your particular development environment and project How to make requirements testable using fit criteria Checklists to help identify stakeholders, users, non-functional requirements, and more requirements patterns New features include Strategy guides for different environments, including outsourcing Strategies for gathering and implementing requirements for iterative releases "Thinking above the line" to find the real problem How to move from requirements to find model for clearer viewpoints of the system Using story cards as requirements Using the Volere Knowledge Model to help record and communicate requirements Fundamental truths about requirements and system development

A Rigorous and Practical Approach, Third Edition

Metainformatics

Software Configuration Management Handbook, Third Edition

Theory and Practice

Experimentation in Software Engineering

The pervasiveness of software in business makes it crucial that software engineers and developers understand how software development impacts an entire organization. Strategic Software Engineering: An Interdisciplinary Approach presents software engineering as a strategic, business-oriented, interdisciplinary endeavor, rather than simply a technical process, as it has been described in previous publications. The book addresses technical, scientific, and management aspects of software development in a way that is accessible to a wide audience. It provides a detailed, critical review of software development models and processes, followed with a strategic assessment of how process models evolved over time and how to improve them. The authors then focus on the relation between problem-solving techniques and strategies for effectively confronting real-world business problems. They also analyze the impact of interdisciplinary factors on software development, including the role of people and business economics. The book concludes with a brief look at specialized system development. The diverse backgrounds of the authors, encompassing computer science, information systems, technology, and business management, help create this book's integrated approach, which answers the demand for a comprehensive, interdisciplinary outlook encompassing all facets of how software relates to an organization.

Software Engineering: A Methodical Approach (Second Edition) provides a comprehensive, but concise introduction to software engineering. It adopts a methodical approach to solving software engineering problems, proven over several years of teaching, with outstanding results. The book covers concepts, principles, design, construction, implementation, and management issues of software engineering. Each chapter is organized systematically into brief, reader-friendly sections, with itemization of the important points to be remembered. Diagrams and illustrations also sum up the salient points to enhance learning. Additionally, the book includes the author's original methodologies that add clarity and creativity to the software engineering experience. New in the Second Edition are chapters on software engineering projects, management support systems, software engineering frameworks and patterns as a significant building block for the design and construction of contemporary software systems, and emerging software engineering frontiers. The text starts with an introduction of software engineering and the role of the software engineer. The following chapters examine in-depth software analysis, design, development, implementation, and management. Covering object-oriented methodologies and the principles of object-oriented information engineering, the book reinforces an object-oriented approach to the early phases of the software development life cycle. It covers various diagramming techniques and emphasizes object classification and object behavior. The text features comprehensive treatments of: Project management aids that are commonly used in software engineering An overview of the software design phase, including a discussion of the software design process, design strategies, architectural design, interface design, database design, and design and development standards User interface design Operations design Design considerations including system catalog, product documentation, user message management, design for real-time software, design for reuse, system security, and the agile effect Human resource management from a software engineering perspective Software economics Software implementation issues that range from operating environments to the marketing of software Software maintenance legacy systems, and re-engineering This textbook can be used as a one-semester or two-semester course in software engineering, augmented with an appropriate CASE or RAD tool. It emphasizes a practical, methodical approach to software engineering, avoiding an overkill of theoretical calculations where possible. The primary objective is to help students gain a solid grasp of the activities in the software development life cycle to be confident about taking on new software engineering projects.

This book focuses on various topics related to engineering and management of requirements, in particular elicitation, negotiation, prioritisation, and documentation (whether with natural languages or with graphical models). The book provides methods and techniques that help to characterise, in a systematic manner, the requirements of the intended engineering system. It was written with the goal of being adopted as the main text for courses on requirements engineering, or as a strong reference to the topics of requirements in courses with a broader scope. It can also be used in vocational courses, for professionals interested in the software and information systems domain. Readers who have finished this book will be able to: - establish and plan a requirements engineering process within the development of complex engineering systems; - define and identify the types of relevant requirements in engineering projects; - choose and apply the most appropriate techniques to elicit the requirements of a given system; - conduct and manage negotiation and prioritisation processes for the requirements of a given engineering system; - document the requirements of the system under development, either in natural language or with graphical and formal models. Each chapter includes a set of exercises.

A Framework for Managing, Measuring, and Predicting Attributes of Software Development Products and Processes Reflecting the immense progress in the development and use of software metrics in the past decades, Software Metrics: A Rigorous and Practical Approach, Third Edition provides an up-to-date, accessible, and comprehensive introduction to software metrics. Like its popular predecessors, this third edition discusses important issues, explains essential concepts, and offers new approaches for tackling long-standing problems. New to the Third Edition This edition contains new material relevant to object-oriented design, design patterns, model-driven development, and agile development processes. It includes a new chapter on causal models and Bayesian networks and their application to software engineering. This edition also incorporates recent references to the latest software metrics activities, including research results, industrial case studies, and standards. Suitable for a Range of Readers With numerous examples and exercises, this book continues to serve a wide audience. It can be used as a textbook for a software metrics and quality assurance course or as a useful supplement in any software engineering course. Practitioners will appreciate the important results that have previously only appeared in research-oriented publications. Researchers will welcome the material on new results as well as the extensive bibliography of measurement-related information. The book also gives software managers and developers practical guidelines for selecting metrics and planning their use in a measurement program.

Model Driven Engineering Languages and Systems

The Good, the Hype and the Ugly

An Interdisciplinary Approach

Software Maintenance Management

Empirical Methods and Studies in Software Engineering

Getting Requirements Right

Featuring an associated Web page, and consistently combining theory with real-world practical applications, this text includes thought-provoking questions about legal and ethical issues in software engineering.

A superior primer on software testing and quality assurance, from integration to execution and automation This important new work fills the pressing need for a user-friendly text that aims to provide software engineers, software quality professionals, software developers, and students with the fundamental developments in testing theory and common testing practices. Software Testing and Quality Assurance: Theory and Practice equips readers with a solid understanding of: Practices that support the production of quality software Software testing techniques Life-cycle models for requirements, testing, test cases, and test results Process models for units, integration, system, and acceptance testing How to build test teams, including recruiting and retaining test engineers Quality Models, Capability Maturity Model, Testing Maturity Model, and Test Process Improvement Model Expertly balancing theory with practice, and complemented with an abundance of pedagogical tools, including test questions, examples, teaching suggestions, and chapter summaries, this book is a valuable, self-contained tool for professionals and an ideal introductory text for courses in software testing, quality assurance, and software engineering.

Basics of Software Engineering Experimentation is a practical guide to experimentation in a field which has long been underpinned by suppositions, assumptions, speculations and beliefs. It demonstrates to software engineers how Experimental Design and Analysis can be used to validate their beliefs and ideas. The book does not assume its readers have an in-depth knowledge of mathematics, specifying the conceptual essence of the techniques to use in the design and analysis of experiments and keeping the mathematical calculations clear and simple. Basics of Software Engineering Experimentation is practically oriented and is specially written for software engineers, all the examples being based on real and fictitious software engineering experiments.

In this book, the authors of the 20-year best-selling classic Security in Computing take a fresh, contemporary, and powerfully relevant new approach to introducing computer security. Organised around attacks and mitigations, the Pfleegers' new Analyzing Computer Security will attract students' attention by building on the high-profile security failures they may have already encountered in the popular media. Each section starts with an attack description. Next, the authors explain the vulnerabilities that have allowed this attack to occur. With this foundation in place, they systematically present today's most effective countermeasures for blocking or weakening the attack. One step at a time, students progress from attack/problem/harm to solution/protection/mitigation, building the powerful real-world problem solving skills they need to succeed as information security professionals. Analyzing Computer Security addresses crucial contemporary computer security themes throughout, including effective security management and risk analysis; economics and quantitative study; privacy,

ethics, and laws; and the use of overlapping controls. The authors also present significant new material on computer forensics, insiders, human factors, and trust.

Basics of Software Engineering Experimentation

The Production of Quality Software

Agile!

Software Engineering: Theory and Practice: Fourth Edition

Insider Threats in Cyber Security

Software Metrics

This user guide provides step-by-step instructions on how to set up, use, and maintain the statistical business register of the Asian Development Bank. It also outlines the key technical considerations that guided the Asian Development Bank's development of the software. This publication serves as a reference for countries looking to establish statistical business registers of their own using the software solution developed as part of this initiative. Statistical business registers are structured databases that provide information on business establishments and their activities, enabling a country's national statistical system to produce economic statistics and indicators. This helps governments design and implement economic policies that are data-driven.

A concise, engineering-oriented resource that provides practical support to IT professionals and those responsible for the quality of the software or systems they develop Software quality stems from two distinctive, but associated, topics in software engineering: software functional quality and software structural quality.

This book studies the tenets of both of these notions, which focus on the efficiency and value of a design, respectively. It addresses engineering quality on both the application and system levels with attention to information systems (IS) and embedded systems (ES) as well as recent developments. Software Quality

Engineering introduces the basic concepts of quality engineering like the nature of the engineering process, quality models and measurements, and evaluation quality, and provides a step-by-step overview of the application of software quality engineering in commonly recognized phases of the software development process. It also discusses management of software quality engineering processes, with special attention to budget, planning, conflict resolution, and traceability of quality requirements. Targeted at graduate engineering students and software quality specialists, Software Quality Engineering: Provides an analysis of interdependence between software functionality and its quality Includes a list of software quality engineering "to-dos" and models of software quality requirements traceability Covers the practical use of related ISO/IEC JTC1/SC7 standards

Nowadays, societies crucially depend on high-quality software for a large part of their functionalities and activities. Therefore, software professionals, researchers, managers, and practitioners alike have to competently decide what software technologies and products to choose for which purpose. For various reasons, systematic empirical studies employing strictly scientific methods are hardly practiced in software engineering. Thus there is an unquestioned need for developing improved and better-qualified empirical methods, for their application in practice and for dissemination of the results. This book describes different kinds of empirical studies and methods for performing such studies, e.g., for planning, performing, analyzing, and reporting such studies. Actual studies are presented in detail in various chapters dealing with inspections, testing, object-oriented techniques, and component-based software engineering.

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.

Project Management of Large Software-Intensive Systems

Experiences from ESERNET

Strategic Software Engineering

User Guide for ADB Statistical Business Register

Concepts and Operations

Requirements in Engineering Projects

Non-Functional Requirements in Software Engineering presents a systematic and pragmatic approach to `building quality into' software systems. Systems must exhibit software quality attributes, such as accuracy, performance, security and modifiability. However, such non-functional requirements (NFRs) are difficult to address in many projects, even though there are many techniques to meet functional requirements in order to provide desired functionality. This is particularly true since the NFRs for each system typically interact with each other, have a broad impact on the system and may be subjective. To enable developers to systematically deal with a system's diverse NFRs, this book presents the NFR Framework. Structured graphical facilities are offered for stating NFRs and managing them by refining and inter-relating NFRs, justifying decisions, and determining their impact. Since NFRs might not be absolutely achieved, they may simply be satisfied sufficiently (`satisfied'). To reflect this, NFRs are represented as `softgoals', whose interdependencies, such as tradeoffs and synergy, are captured in graphs. The impact of decisions is qualitatively propagated through the graph to determine how well a chosen target system satisfies its NFRs. Throughout development, developers direct the process, using their expertise while being aided by catalogues of knowledge about NFRs, development techniques and tradeoffs, which can all be explored, reused and customized. Non-Functional Requirements in Software Engineering demonstrates the applicability of the NFR Framework to a variety of NFRs, domains, system characteristics and application areas. This will help readers apply the Framework to NFRs and domains of particular interest to them. Detailed treatments of particular NFRs - accuracy, security and performance requirements - along with treatments of NFRs for information systems are presented as specializations of the NFR Framework. Case studies of NFRs for a variety of information systems include credit card and administrative systems. The use of the Framework for particular application areas is illustrated for software architecture as well as enterprise modelling. Feedback from domain experts in industry and government provides an initial evaluation of the Framework and some case studies. Drawing on research results from several theses and refereed papers, this book's presentation, terminology and graphical notation have been integrated and illustrated with many figures. Non-Functional Requirements in Software Engineering is an excellent resource for software engineering practitioners, researchers and students.

Insider Threats in Cyber Security is a cutting edge text presenting IT and non-IT facets of insider threats together. This volume brings together a critical mass of well-established worldwide researchers, and provides a unique multidisciplinary overview. Monica van Huystee, Senior Policy Advisor at MCI, Ontario, Canada comments "The book will be a must read, so of course I'll need a copy." Insider Threats in Cyber Security covers all aspects of insider threats, from motivation to mitigation. It includes how to monitor insider threats (and what to monitor for), how to mitigate insider threats, and related topics and case studies. Insider Threats in Cyber Security is intended for a professional audience composed of the military, government policy makers and banking; financing companies focusing on the Secure Cyberspace industry. This book is also suitable for advanced-level students and researchers in computer science as a secondary text or reference book.

The most comprehensive General, Organic, and Biochemistry book available, Introduction to General, Organic, and Biochemistry, 11th Edition continues its tradition of a solid development of problem-solving skills, numerous examples and practice problems, along with coverage of current applications. Written by an experienced author team, they skillfully anticipate areas of difficulty and pace the book accordingly. Readers will find the right mix of general chemistry compared to the discussions on organic and biochemistry. Introduction to General, Organic, and Biochemistry, 11th Edition has clear & logical explanations of chemical concepts and great depth of coverage as well as a clear, consistent writing style which provides great readability. An emphasis on Real-World aspects of chemistry makes the reader comfortable in seeing how the chemistry will apply to their career.

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

A Practitioner's Approach

Guide to Advanced Empirical Software Engineering

Non-Functional Requirements in Software Engineering

Fourth International Baltic Workshop, Baltic DB&IS 2000 Vilnius, Lithuania, May 1–5, 2000 Selected Papers

Software Quality Assurance

Software Testing

A Student Guide to Object-Oriented Development is an introductory text that follows the software development process, from requirements capture to implementation, using an object-oriented approach. The book uses object-oriented techniques to present a practical viewpoint on developing software, providing the reader with a basic understanding of object-oriented concepts by developing the subject in an uncomplicated and easy-to-follow manner. It is based on a main worked case study for teaching purposes, plus others with password-protected answers on the web for use in coursework or exams. Readers can benefit from the authors' years of teaching experience. The book outlines standard object-oriented modelling techniques and illustrates them with a variety of examples and exercises, using UML as the modelling language and Java as the language of implementation. It adopts a simple, step by step approach to object-oriented development, and includes case studies, examples, and exercises with solutions to consolidate learning. There are 13 chapters covering a variety of topics such as sequence and collaboration diagrams; state diagrams; activity diagrams; and implementation diagrams. This book is an ideal reference for students taking undergraduate introductory/intermediate computing and information systems courses, as well as business studies courses and conversion masters' programmes. Adopts a simple, step by step approach to object-oriented development Includes case studies, examples, and exercises with solutions to consolidate learning Benefit from the authors' years of teaching experience

In recent years, our world has experienced a profound shift and progression in available computing and knowledge sharing innovations. These emerging advancements have developed at a rapid pace, disseminating into and affecting numerous aspects of contemporary society. This has created a pivotal need for an innovative compendium encompassing the latest trends, concepts, and issues surrounding this relevant discipline area. During the past 15 years, the Encyclopedia of Information Science and Technology has become recognized as one of the landmark sources of the latest knowledge and discoveries in this discipline. The Encyclopedia of Information Science and Technology, Fourth Edition is a 10-volume set which includes 705 original and previously unpublished research articles covering a full range of perspectives, applications, and techniques contributed by thousands of experts and researchers from around the globe. This authoritative encyclopedia is an all-encompassing, well-established reference source that is ideally designed to disseminate the most forward-thinking and diverse research findings. With critical perspectives on the impact of information science management and new technologies in modern settings, including but not limited to computer science, education, healthcare, government, engineering, business, and natural and physical sciences, it is a pivotal and relevant source of knowledge that will benefit every professional within the field of information science and technology and is an invaluable addition to every academic and corporate library.

Mastering the Requirements Process

Analyzing Computer Security

Databases and Information Systems

Software Quality Engineering

What Should the Department of Defense's Role in Cyber Be?