

Title Software Engineering 7th Edition Author Ian

Software engineering requires specialized knowledge of a broad spectrum of topics, including the construction of software and the platforms, applications, and environments in which the software operates as well as an understanding of the people who build and use the software. Offering an authoritative perspective, the two volumes of the Encyclopedia of Software Engineering cover the entire multidisciplinary scope of this important field. More than 200 expert contributors and reviewers from industry and academia across 21 countries provide easy-to-read entries that cover software requirements, design, construction, testing, maintenance, configuration management, quality control, and software engineering management tools and methods. Editor Phillip A. Laplante uses the most universally recognized definition of the areas of relevance to software engineering, the Software Engineering Body of Knowledge (SWEBOK®), as a template for organizing the material. Also available in an electronic format, this encyclopedia supplies software engineering students, IT professionals, researchers,

managers, and scholars with unrivaled coverage of the topics that encompass this ever-changing field. Also Available Online This Taylor & Francis encyclopedia is also available through online subscription, offering a variety of extra benefits for researchers, students, and librarians, including: Citation tracking and alerts Active reference linking Saved searches and marked lists HTML and PDF format options Contact Taylor and Francis for more information or to inquire about subscription options and print/online combination packages. US: (Tel) 1.888.318.2367; (E-mail) e-reference@taylorandfrancis.com International: (Tel) +44 (0) 20 7017 6062; (E-mail) online.sales@tandf.co.uk

An introduction to a broad range of topics in deep learning, covering mathematical and conceptual background, deep learning techniques used in industry, and research perspectives. "Written by three experts in the field, Deep Learning is the only comprehensive book on the subject." –Elon Musk, cochair of OpenAI; cofounder and CEO of Tesla and SpaceX Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of

concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator to formally specify all the knowledge that the computer needs. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones; a graph of these hierarchies would be many layers deep. This book introduces a broad range of topics in deep learning. The text offers mathematical and conceptual background, covering relevant concepts in linear algebra, probability theory and information theory, numerical computation, and machine learning. It describes deep learning techniques used by practitioners in industry, including deep feedforward networks, regularization, optimization algorithms, convolutional networks, sequence modeling, and practical methodology; and it surveys such applications as natural language processing, speech recognition, computer vision, online recommendation systems, bioinformatics, and videogames. Finally, the book offers research perspectives, covering such theoretical topics as linear factor models, autoencoders, representation learning, structured probabilistic models, Monte Carlo methods, the partition function, approximate

inference, and deep generative models. Deep Learning can be used by undergraduate or graduate students planning careers in either industry or research, and by software engineers who want to begin using deep learning in their products or platforms. A website offers supplementary material for both readers and instructors.

For over 20 years, Software Engineering: A Practitioner's Approach has been the best selling guide to software engineering for students and industry professionals alike. The sixth edition continues to lead the way in software engineering. A new Part 4 on Web Engineering presents a complete engineering approach for the analysis, design, and testing of Web Applications, increasingly important for today's students. Additionally, the UML coverage has been enhanced and significantly increased in this new edition. The pedagogy has also been improved in the new edition to include sidebars. They provide information on relevant software tools, specific work flow for specific kinds of projects, and additional information on various topics. Additionally, Pressman provides a running case study called "Safe Home" throughout the book, which provides the application

of software engineering to an industry project. New additions to the book also include chapters on the Agile Process Models, Requirements Engineering, and Design Engineering. The book has been completely updated and contains hundreds of new references to software tools that address all important topics in the book. The ancillary material for the book includes an expansion of the case study, which illustrates it with UML diagrams. The On-Line Learning Center includes resources for both instructors and students such as checklists, 700 categorized web references, Powerpoints, a test bank, and a software engineering library-containing over 500 software engineering papers. TAKEAWY HERE IS THE FOLLOWING:1. AGILE PROCESS METHODS ARE COVERED EARLY IN CH. 42. NEW PART ON WEB APPLICATIONS --5 CHAPTERS

This book covers two applications of ontologies in software engineering and software technology: sharing knowledge of the problem domain and using a common terminology among all stakeholders; and filtering the knowledge when defining models and metamodels. By presenting the advanced use of ontologies in software research and software projects, this book is of benefit to software engineering researchers in both academia and

industry.

***Client-Centered Software Development
Software Engineering
Version 3.0***

Guide to the Software Engineering Body of Knowledge (Swebok(r))

This Seventh Edition of Donald Reifer's popular, bestselling tutorial summarizes what software project managers need to know to be successful on the job. The text provides pointers and approaches to deal with the issues, challenges, and experiences that shape their thoughts and performance. To accomplish its goals, the volume explores recent advances in dissimilar fields such as management theory, acquisition management, globalization, knowledge management, licensing, motivation theory, process improvement, organization dynamics, subcontract management, and technology transfer. Software Management provides software managers at all levels of the organization with the information they need to know to develop their software engineering management strategies for now and the future. The book provides insight into management tools and techniques that work in practice. It also provides sufficient instructional materials to serve as a text for a course in software management. This new edition

achieves a balance between theory and practical experience. Reifer systematically addresses the skills, knowledge, and abilities that software managers, at any level of experience, need to have to practice their profession effectively. This book contains original articles by leaders in the software management field written specifically for this tutorial, as well as a collection of applicable reprints. About forty percent of the material in this edition has been produced specifically for the tutorial. Contents: * Introduction * Life Cycle Models * Process Improvement * Project Management * Planning Fundamentals * Software Estimating * Organizing for Success * Staffing Essentials * Direction Advice * Visibility and Control * Software Risk Management * Metrics and Measurement * Acquisition Management * Emerging Management Topics "The challenges faced by software project managers are the gap between what the customers can envision and the reality on the ground and how to deal with the risks associated with this gap in delivering a product that meets requirements on time and schedule at the target costs. This tutorial hits the mark by providing project managers, practitioners, and educators with source materials on how project managers can effectively deal with this risk." -Dr. Kenneth E. Nidiffer, Systems & Software Consortium, Inc. "The volume has evolved into a solid set of foundation works for anyone trying to practice software management in

a world that is increasingly dependent on software release quality, timeliness, and productivity." -Walker Royce, Vice President, IBM Software Services-Rational

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 10th 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. The full text downloaded to your computer With eBooks you can: search for key concepts, words and phrases make highlights and notes as you study share your notes with friends eBooks are downloaded to your computer and accessible either offline through the Bookshelf (available as a

free download), available online and also via the iPad and Android apps. Upon purchase, you'll gain instant access to this eBook. Time limit The eBooks products do not have an expiry date. You will continue to access your digital ebook products whilst you have your Bookshelf installed.

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

Refined and streamlined, SYSTEMS ANALYSIS AND DESIGN IN A CHANGING WORLD, 7E helps students develop the conceptual, technical, and managerial foundations for systems analysis design and implementation as well as project management principles for systems development. Using case driven techniques, the succinct 14-chapter text focuses on content that is key for success in today's market. The authors' highly effective presentation teaches both traditional (structured) and object-oriented (OO) approaches to systems analysis and design. The book highlights use cases, use diagrams, and use case descriptions required for a modeling approach, while demonstrating their application to traditional, web development, object-oriented, and service-oriented architecture approaches. The Seventh Edition's refined sequence of topics makes it easier to read and understand than ever. Regrouped analysis and design chapters provide more flexibility in course organization. Additionally, the text's running cases have been completely updated and now include a stronger focus on connectivity in applications. Important Notice: Media content referenced within the product description or the product text may not be available in the ebook version.

Software Engineering, Global Edition

Deep Learning

Encyclopedia of Software Engineering Three-Volume Set (Print)

Coding Literacy

Software Design - Cognitive Aspect

The latest title in Addison Wesley's world-renowned Robert C. Martin Series on better software development, *Code That Fits in Your Head* offers indispensable practical advice for writing code at a sustainable pace, and controlling the complexity that causes too many software projects to spin out of control. Reflecting decades of experience consulting on software projects and helping development teams succeed, Mark Seemann shares proven practices and heuristics, supported by realistic advice. His guidance ranges from checklists to teamwork, encapsulation to decomposition, API design to unit testing and troubleshooting. Throughout, Seemann illuminates his insights with up-to-date code examples drawn from a start to finish sample project. Seemann's examples are written in C##, and designed to be clear and useful to every object-oriented enterprise developer, whether they use C#, Java, or another language. *Code That Fits in Your Head* is accompanied by the complete code base for this sample application, organized in a Git repository to facilitate further exploration of details that don't fit in the text. *PMBOK® Guide* is the go-to resource for project management practitioners. The project management profession has significantly evolved due to emerging technology, new approaches and rapid market changes. Reflecting this evolution, *The Standard for Project Management* enumerates 12 principles of project management and the *PMBOK® Guide* &– Seventh

Edition is structured around eight project performance domains. This edition is designed to address practitioners' current and future needs and to help them be more proactive, innovative and nimble in enabling desired project outcomes. This edition of the PMBOK® Guide:

- Reflects the full range of development approaches (predictive, adaptive, hybrid, etc.);
- Provides an entire section devoted to tailoring the development approach and processes;
- Includes an expanded list of models, methods, and artifacts;
- Focuses on not just delivering project outputs but also enabling outcomes; and
- Integrates with PMI standards+™ for information and standards application content based on project type, development approach, and industry sector.

Classical and Object-Oriented Software Engineering, 5/e is designed for an introductory software engineering course. This book provides an excellent introduction to software engineering fundamentals, covering both traditional and object-oriented techniques. Schach's unique organization and style makes it excellent for use in a classroom setting. It presents the underlying software engineering theory in Part I and follows it up with the more practical life-cycle material in Part II. Many software engineering books are more like reference books, which do not provide the appropriate fundamentals before inundating students with implementation details. In this edition, more practical material has been added to help students understand how to use what they are learning. This has been done through the use of "How To" boxes and greater implementation detail in the case study. Additionally, the new edition contains the references to the most current literature and includes an overview of extreme programming. The website in

this edition will be more extensive. It will include Solutions, PowerPoints that incorporate lecture notes, newly developed self-quizz questions, and source code for the term project and case study.

Lean Software Development: An Agile Toolkit Adapting agile practices to your development organization Uncovering and eradicating waste throughout the software development lifecycle Practical techniques for every development manager, project manager, and technical leader Lean software development: applying agile principles to your organization In Lean Software Development, Mary and Tom Poppendieck identify seven fundamental "lean" principles, adapt them for the world of software development, and show how they can serve as the foundation for agile development approaches that work. Along the way, they introduce 22 "thinking tools" that can help you customize the right agile practices for any environment. Better, cheaper, faster software development. You can have all three—if you adopt the same lean principles that have already revolutionized manufacturing, logistics and product development. Iterating towards excellence: software development as an exercise in discovery Managing uncertainty: "decide as late as possible" by building change into the system. Compressing the value stream: rapid development, feedback, and improvement Empowering teams and individuals without compromising coordination Software with integrity: promoting coherence, usability, fitness, maintainability, and adaptability How to "see the whole"—even when your developers are scattered across multiple locations and contractors Simply put, Lean Software Development helps you refocus development on value, flow, and people—so you can achieve breakthrough

quality, savings, speed, and business alignment.

Computer Science Illuminated

PHP & MySQL: Novice to Ninja

Agile Management for Software Engineering Complete Self-Assessment Guide

Software Engineering: A Practitioner's Approach

Green in Software Engineering

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.

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.

In business, driving value is a key strategy and typically starts at the top of an organization. In today ' s digital age, driving software value is also an important, and often overlooked, key strategy. Executives, and the corporate board, need

to expect the highest level of business value from the software the organization is developing, buying, and selling. In today ' s digital transformation marketplace, it is imperative that organizations start driving business value from software development initiatives. For many years, the cost of software development challenged organizations with questions such as: How do we allocate software development costs? Should these costs be considered an overhead expense? Are we getting the most value possible for our investment? A fundamental problem has been built into these questions – the focus on cost. In almost every other part of the organization, maximizing profit or, in the case of a not-for-profit, maximizing the funds available, provides a clear focus with metrics to determine success or failure. In theory, simply aligning software spending with the maximizing profit goals should be sufficient to avoid any questions about value for money. Unfortunately, this alignment hasn ' t turned out to be so simple, and the questions persist, particularly at the strategic or application portfolio level. In this book, Michael D.S. Harris describes how a software business value culture—one where all stakeholders, including technology and business—have a clear understanding of the goals and expected business value from software development. The book shows readers how they can transform software development from a cost or profit center to a business value center. Only a culture of software as a value center enables an

organization to constantly maximize business value flow through software development. If your organization is starting to ask how it can change software from a cost-center to a value-center, this book is for you.

Covering a variety of areas including software analysis, design, coding and maintenance, this text details the research conducted since the 1970s in this fast-developing field before going on to define a computer program from the viewpoint of computing and cognitive psychology. The two essential sides of programming, software production and software understanding, are given detailed treatment, with parallels drawn throughout between studies on processing texts written in natural language and processing computer programs. Of particular interest to researchers, practitioners and graduates in cognitive psychology, cognitive ergonomics and computer science.

Head First Software Development

A Practitioner's Approach

From Empirical Studies to Open Source Artifacts

Ontologies for Software Engineering and Software Technology

Systems Analysis and Design in a Changing World

This guide offers students an overview of computer science principles, and provides a solid foundation for those continuing their study in this dynamic and exciting

discipline. New features of this edition include: a chapter on computer security providing readers with the latest information on preventing unauthorized access; types of malware and anti-virus software; protecting online information, including data collection issues with Facebook, Google, etc.; security issues with mobile and portable devices; a new section on cloud computing offering readers an overview of the latest way in which businesses and users interact with computers and mobile devices; a rewritten section on social networks including new data on Google+ and Facebook; updates to include HTML5; revised and updated Did You Know callouts are included in the chapter margins; revisions of recommendations by the ACM dealing with computer ethic issues. --

What others in the trenches say about The Pragmatic Programmer... "The cool thing about this book is that it's great for keeping the programming process fresh. The book helps you to continue to grow and clearly comes from people who have been there." -Kent Beck, author of Extreme

Programming Explained: Embrace Change “I found this book to be a great mix of solid advice and wonderful analogies!”
–Martin Fowler, author of Refactoring and UML Distilled “I would buy a copy, read it twice, then tell all my colleagues to run out and grab a copy. This is a book I would never loan because I would worry about it being lost.” –Kevin Ruland, Management Science, MSG-Logistics “The wisdom and practical experience of the authors is obvious. The topics presented are relevant and useful.... By far its greatest strength for me has been the outstanding analogies–tracer bullets, broken windows, and the fabulous helicopter-based explanation of the need for orthogonality, especially in a crisis situation. I have little doubt that this book will eventually become an excellent source of useful information for journeymen programmers and expert mentors alike.” –John Lakos, author of Large-Scale C++ Software Design “This is the sort of book I will buy a dozen copies of when it comes out so I can give it to my clients.” –Eric Vought, Software Engineer “Most modern books on software development fail to

cover the basics of what makes a great software developer, instead spending their time on syntax or technology where in reality the greatest leverage possible for any software team is in having talented developers who really know their craft well. An excellent book.” –Pete McBreen, Independent Consultant “Since reading this book, I have implemented many of the practical suggestions and tips it contains. Across the board, they have saved my company time and money while helping me get my job done quicker! This should be a desktop reference for everyone who works with code for a living.” –Jared Richardson, Senior Software Developer, iRenaissance, Inc. “I would like to see this issued to every new employee at my company...” –Chris Cleeland, Senior Software Engineer, Object Computing, Inc. “If I’m putting together a project, it’s the authors of this book that I want. . . . And failing that I’d settle for people who’ve read their book.” –Ward Cunningham Straight from the programming trenches, *The Pragmatic Programmer* cuts through the increasing specialization and technicalities of modern

software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to Fight software rot; Avoid the trap of duplicating knowledge; Write flexible, dynamic, and adaptable code; Avoid programming by coincidence; Bullet-proof your code with contracts, assertions, and exceptions; Capture real requirements; Test ruthlessly and effectively; Delight your users; Build teams of pragmatic programmers; and Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, *The Pragmatic Programmer* illustrates the best practices and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects,

use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer.

Widely considered one of the best practical guides to programming, Steve McConnell's original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help

you: Design for minimum complexity and maximum creativity
Reap the benefits of collaborative development Apply
defensive programming techniques to reduce and flush out
errors Exploit opportunities to refactor—or evolve—code, and
do it safely Use construction practices that are right-
weight for your project Debug problems quickly and
effectively Resolve critical construction issues early and
correctly Build quality into the beginning, middle, and end
of your project

For courses in computer science and software engineering The
Fundamental Practice of Software Engineering Software
Engineering introduces readers 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 readers 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.

Lessons Learned from Programming Over Time

Occupational Outlook Handbook

Object-oriented Software Engineering

A Comprehensive Guide to Software Development Projects

Code That Fits in Your Head

Provides information on successful software development, covering such topics as customer requirements, task estimates, principles of good design, dealing with source code, system testing, and handling bugs.

In the Guide to the Software Engineering Body of Knowledge (SWEBOK(R) Guide), the IEEE Computer Society establishes a baseline for the body of knowledge for the field of software engineering, and the work supports the

Society's responsibility to promote the advancement of both theory and practice in this field. It should be noted that the Guide does not purport to define the body of knowledge but rather to serve as a compendium and guide to the knowledge that has been developing and evolving over the past four decades. Now in Version 3.0, the Guide's 15 knowledge areas summarize generally accepted topics and list references for detailed information. The editors for Version 3.0 of the SWEBOOK(R) Guide are Pierre Bourque (Ecole de technologie superieure (ETS), Universite du Quebec) and Richard E. (Dick) Fairley (Software and Systems Engineering Associates (S2EA)).

How the theoretical tools of literacy help us understand programming in its historical, social and conceptual contexts. The message from educators, the tech community, and even politicians is clear: everyone should learn to code. To emphasize the universality and importance of computer programming, promoters of coding for everyone often invoke the concept of "literacy," drawing parallels between reading and writing code and reading and writing text. In this book, Annette Vee examines the coding-as-literacy analogy and argues that it can be an apt rhetorical frame. The theoretical tools of literacy help us understand programming beyond a technical level, and in its historical, social, and conceptual contexts. Viewing programming from the perspective of literacy and literacy from the perspective of programming, she argues, shifts our

understandings of both. Computer programming becomes part of an array of communication skills important in everyday life, and literacy, augmented by programming, becomes more capacious. Vee examines the ways that programming is linked with literacy in coding literacy campaigns, considering the ideologies that accompany this coupling, and she looks at how both writing and programming encode and distribute information. She explores historical parallels between writing and programming, using the evolution of mass textual literacy to shed light on the trajectory of code from military and government infrastructure to large-scale businesses to personal use. Writing and coding were institutionalized, domesticated, and then established as a basis for literacy. Just as societies demonstrated a “literate mentality” regardless of the literate status of individuals, Vee argues, a “computational mentality” is now emerging even though coding is still a specialized skill.

Software Engineering A Practitioners Approach

A Practitioners Approach

Clean Code

A Guide to the Project Management Body of Knowledge (PMBOK® Guide) –

Seventh Edition and The Standard for Project Management (RUSSIAN)

Practical Software Development Using UML and Java

Get Up to Speed With PHP the Easy Way

PHP & MySQL: Novice to Ninja, 6th Edition is a hands-on guide to learning all the tools, principles, and techniques needed to build a fully functional application using PHP & MySQL. Comprehensively updated to cover PHP 7 and modern best practice, this practical and fun book covers everything from installing PHP and MySQL through to creating a complete online content management system. You'll learn how to: Install PHP & MySQL on Windows, Mac OS X, or Linux Gain a thorough understanding of PHP syntax Use object oriented programming techniques Master database design principles and SQL Develop robust websites that can handle high levels of traffic Build a working content management system (CMS) And much more!

For almost three decades, Roger Pressman's Software Engineering: A Practitioner's Approach has been the world's leading textbook in software engineering. The new eighth edition represents a major restructuring and update of previous editions, solidifying the book's position as the most comprehensive guide to this important subject. The eighth edition of Software Engineering: A Practitioner's Approach has been designed to consolidate and restructure the content introduced over the past two editions of the book. The chapter structure will return to a more linear presentation of software engineering topics with a direct emphasis on the major activities that are part of a generic software process. Content will focus on widely used software engineering methods and will de-emphasize or completely eliminate discussion of secondary methods, tools and techniques. The intent is to provide a more targeted, prescriptive, and focused approach, while attempting to maintain SEPA's reputation as a comprehensive guide to software engineering. The 39 chapters of the eighth edition are organized into five parts - Process, Modeling, Quality Management, Managing Software

Projects, and Advanced Topics. The book has been revised and restructured to improve pedagogical flow and emphasize new and important software engineering processes and practices.

Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of heuristics and "smells" accumulated from the process of writing clean code.

For over 20 years, this has been the best-selling guide to software engineering for students and industry professionals alike. This seventh edition features a new part four on web engineering, which presents a complete engineering approach for the analysis, design and testing of web applications.

Object-Oriented and Classical Software Engineering

Software Engineering at Google

How Computer Programming Is Changing Writing

The Business Value of Software

From Journeyman to Master

This is the eBook of the printed book and may not include any media, website access codes, or print supplements that may come 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

This is the first book that presents a comprehensive overview of sustainability aspects in software engineering. Its format follows the structure of the SWEBOK and covers the key areas involved in the incorporation of green aspects in software engineering, encompassing topics from requirement elicitation to quality assurance and maintenance, while also considering professional practices and economic aspects. The book consists of thirteen chapters, which are structured in five parts. First the

“ Introduction ” gives an overview of the primary general concepts related to Green IT, discussing what Green in Software Engineering is and how it differs from Green by Software Engineering. Next “ Environments, Processes and Construction ” presents green software development environments, green software engineering processes and green software construction in general. The third part, “ Economic and Other Qualities, ” details models for measuring how well software supports green software engineering techniques and for performing trade-off analyses between alternative green practices from an economic perspective. “ Software Development Process ” then details

techniques for incorporating green aspects at various stages of software development, including requirements engineering, design, testing, and maintenance. In closing, “ Practical Issues ” addresses the repercussions of green software engineering on decision-making, stakeholder participation and innovation management. The audience for this book includes software engineering researchers in academia and industry seeking to understand the challenges and impact of green aspects in software engineering, as well as practitioners interested in learning about the state of the art in Green in Software Engineering.

This is an introductory book to information modelling with UML, for entry level university students. It assumes no previous knowledge of UML on the part of the reader, and uses a case-based approach to present the material clearly and accessibly. It harmonises the UML notation with a full software development approach, from project conception through to testing, deployment and enhancement. The author is an experienced tutor, who also practices as a UML professional, and the cases are based upon his own experience. The book is accompanied by a website that provides solutions to end-of-chapter exercises, a password-protected tutor's file of further exercises with solutions, slides to accompany the book, and other support material. This book is suitable for all undergraduate computing and information systems, or Software Engineering courses. First year students will find it particularly helpful for modules on systems development or analysis and design.

For almost four decades, Software Engineering: A Practitioner's Approach (SEPA) has been the world's leading textbook in software engineering. The ninth edition represents a major restructuring and update of previous editions, solidifying the book's position as the most comprehensive guide to this important subject.

Lean Software Development

An Agile Toolkit: An Agile Toolkit

The Art of Failure

Software Development with UML

The CO-FOSS Approach

This book covers the essential knowledge and skills needed by a student who is specializing in software engineering. Readers will learn principles of object orientation, software development, software modeling, software design, requirements analysis, and testing. The use of the Unified Modelling Language to develop software is taught in depth. Many concepts are illustrated using complete examples, with code written in Java.

An exploration of why we play video games despite the fact that we are almost certain to feel unhappy when we fail at them. We may think of video games as being "fun," but in The Art of Failure, Jesper Juul claims that this is almost entirely mistaken. When we play video games, our facial expressions are rarely those of happiness or bliss. Instead, we frown, grimace, and shout

in frustration as we lose, or die, or fail to advance to the next level. Humans may have a fundamental desire to succeed and feel competent, but game players choose to engage in an activity in which they are nearly certain to fail and feel incompetent. So why do we play video games even though they make us unhappy? Juul examines this paradox. In video games, as in tragic works of art, literature, theater, and cinema, it seems that we want to experience unpleasantness even if we also dislike it. Reader or audience reaction to tragedy is often explained as catharsis, as a purging of negative emotions. But, Juul points out, this doesn't seem to be the case for video game players. Games do not purge us of unpleasant emotions; they produce them in the first place. What, then, does failure in video game playing do? Juul argues that failure in a game is unique in that when you fail in a game, you (not a character) are in some way inadequate. Yet games also motivate us to play more, in order to escape that inadequacy, and the feeling of escaping failure (often by improving skills) is a central enjoyment of games. Games, writes Juul, are the art of failure: the singular art form that sets us up for failure and allows us to experience it and experiment with it. *The Art of Failure* is essential reading for anyone interested in video games, whether as entertainment, art, or education.

Client-Centered Software Development: The CO-FOSS Approach introduces a

method to creating a customized software product for a single client, either from scratch or by reusing open source components. The clients are typically non-profit humanitarian, educational, or public service organizations. This approach has been used in undergraduate courses where students learn the principles of software development while implementing a real-world software product. This book provides instructors, students, clients, and professional software developers with detailed guidance for developing a new CO-FOSS product from conceptualization to completion. Features Provides instructors, students, clients, and professional software developers with a roadmap for the development of a new CO-FOSS product from conceptualization to completion Motivates students with real-world projects and community service experiences Teaches all elements of the software process, including requirements gathering, design, collaboration, coding, testing, client communication, refactoring, and writing developer and user documentation Uses source code that can be reused and refitted to suit the needs of future projects, since each CO-FOSS product is free and open source software Provides links to a rich variety of resources for instructors and students to freely use in their own courses that develop new CO-FOSS products for other non-profits.

Research and Evidence in Software Engineering: From Empirical Studies to

Open Source Artifacts introduces advanced software engineering to software engineers, scientists, postdoctoral researchers, academicians, software consultants, management executives, doctoral students, and advanced level postgraduate computer science students. This book contains research articles addressing numerous software engineering research challenges associated with various software development-related activities, including programming, testing, measurements, human factors (social software engineering), specification, quality, program analysis, software project management, and more. It provides relevant theoretical frameworks, empirical research findings, and evaluated solutions addressing the research challenges associated with the above-mentioned software engineering activities. To foster collaboration among the software engineering research community, this book also reports datasets acquired systematically through scientific methods and related to various software engineering aspects that are valuable to the research community. These datasets will allow other researchers to use them in their research, thus improving the quality of overall research. The knowledge disseminated by the research studies contained in the book will hopefully motivate other researchers to further innovation in the way software development happens in real practice.

Software Design

Software Engineering for Internet Applications

Heuristics for Software Engineering

Software Management

An Essay on the Pain of Playing Video Games

This book is perhaps the first attempt to give full treatment to the topic of Software Design.

It will facilitate the academia as well as the industry. This book covers all the topics of software design including the ancillary ones.

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.

A Handbook of Agile Software Craftsmanship

Research and Evidence in Software Engineering

Code Complete

American Book Publishing Record

The Pragmatic Programmer