Software Engineering By Technical Publications

This book offers a primer on the valuation of digital intangibles, a trending class of immaterial assets. Startups like successful unicorns, as well as consolidated firms desperately working to re-engineer their business models, are now trying to go digital and to reap higher returns by exploiting new intangibles. This book is innovative in its design and concept since it tackles a frontier topic with an original methodology, combining academic rigor with practical insights. Digital intangibles range from digitized versions of traditional immaterial assets (brands, patents, know-how, etc.) to more trendy applications like big data, Internet of Things, interoperable databases, artificial intelligence, digital newspapers, social networks, blockchains, FinTech applications, etc. This book comprehensively addresses related valuation issues, and demonstrates how best practices can be applied to specific asset appraisals, making it of interest to researchers, students, and practitioners alike.

Innovations in software engineering have ushered in an era of wired technology. We are constantly surrounded by the products of this revolution. With this book, the author has created a resourceful cache of latest information for aspiring software engineers, preparing them for a productive industry experience. Elaboration on concepts of software development and engineering, the book gives an insightful view of the fundamentals of system design, coding and documentation, software metrics, management and cost estimation. Based upon the updated university curriculum, this book is a student-friendly work that explains difficult concepts with neat illustrations and examples. Topic wise discussions on system testing and computer-aided software engineering go a long way in equipping budding software engineers with the right knowledge and expertise. This is a great book for self-based learning and for competitive examinations. It comes with a glossary of technical terms. Key Features • Lucid, well-explained concepts with solved examples • Complete coverage of the updated university syllabus • Chapter-end summaries and questions for quick review • Relevant illustrations for better understanding and retention • Glossary of technical terms • Solution to previous years' university papers

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 Software architecture is foundational to the development of large, practical software-intensive applications. This brand-new text covers all facets of software architecture and how it serves as the intellectual centerpiece of software development and evolution. Critically, this text focuses on supporting creation of real implemented systems. Hence the text details not only modeling techniques, but design, implementation, deployment, and system adaptation -- as well as a host of other topics -- putting the elements in context and comparing and contrasting them with one another. Rather than focusing on one method, notation, tool, or process, this new text/reference widely surveys software architecture techniques, enabling the instructor and practitioner to choose the right tool for the job at hand. Software Architecture is intended for upper-division undergraduate and graduate courses in software architecture, software design, component-based software engineering, and distributed systems; the text may also be used in introductory as well as advanced software engineering courses.

Software EngineeringTechnical PublicationsSoftware EngineeringAdvanced JavaTechnical Publications

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

Taking a learn-by-doing approach, Software Engineering Design: Theory and Practice uses examples, review questions, chapter exercises, and case study assignments to provide students and practitioners with the understanding required to design complex software systems. Explaining the concepts that are immediately relevant to software designers, it be

Advanced Java is a textbook specially designed for undergraduate and post graduate students of Computer Science. It focuses on developing the applications both at basic and moderate level. This text book is divided into seven units. The first unit introduces Java network programming. In this unit along with the basic concepts of networking, the programming using Sockets, InetAddress, URL and URLConnection class is discussed in a lucid manner. The second unit is based on JDBC programming. In this unit, connecting with the database is discussed with examples and illustrations. Then next two chapters focuses on server side programming by means of Servlet programming and JSP. In third unit, the illustration of how to create and execute servlets is given. Then the concept of cookies and session management is discussed. In the next subsequent unit the Java Server Pages - its overview and programming is studied. In the last three units the advanced concepts of Java programming such as JSF, Hibernate and Java Web Framework : Spring is discussed. The contents of this textbook is supported with numerous illustrations, examples, program codes, and screenshots. With its lucid presentation and inclusion of numerous examples the book will be very useful for the readers.

The practice of enterprise application development has benefited from the emergence of many new enabling technologies. Multi-tiered object-oriented platforms, such as Java and .NET, have become commonplace. These new tools and technologies are capable of building powerful applications, but they are not easily implemented. Common failures in enterprise applications often occur because their developers do not understand the architectural lessons that experienced object developers have learned. Patterns of Enterprise Application Architecture is written in direct response to the stiff challenges that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology--from Smalltalk to CORBA to Java to .NET--the same basic design ideas can be adapted and applied to solve common problems. With the help of an expert group of contributors, Martin distills over forty recurring solutions into patterns. The result is an indispensable handbook of solutions that are applicable to any enterprise application platform. This book is actually

two books in one. The first section is a short tutorial on developing enterprise applications, which you can read from start to finish to understand the scope of the book's lessons. The next section, the bulk of the book, is a detailed reference to the patterns themselves. Each pattern provides usage and implementation information, as well as detailed code examples in Java or C#. The entire book is also richly illustrated with UML diagrams to further explain the concepts. Armed with this book, you will have the knowledge necessary to make important architectural decisions about building an enterprise application and the proven patterns for use when building them. The topics covered include · Dividing an enterprise application into layers · The major approaches to organizing business logic · An in-depth treatment of mapping between objects and relational databases · Using Model-View-Controller to organize a Web presentation · Handling concurrency for data that spans multiple transactions · Designing distributed object interfaces

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 signficantly increased in this new edition. The pedagogy has also been improved in the new edition to include sidebars. They provide information on relevant softare 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

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

Based on the results of the study carried out in 1996 to investigate the state of the art of workflow and process technology, MCC initiated the Collaboration Management Infrastructure (CMI) research project to develop innovative agent-based process technology that can support the process requirements of dynamically changing organizations and the requirements of nomadic computing. With a research focus on the flow of interaction among people and software agents representing people, the project deliverables will include a scalable, heterogeneous, ubiquitous and nomadic infrastructure for business processes. The resulting technology is being tested in applications that stress an intensive mobile collaboration among people as part of large, evolving business processes. Workflow and Process Automation: Concepts and Technology provides an overview of the problems and issues related to process and workflow technology, and in particular to definition and analysis of processes and workflows, and execution of their instances. The need for a transactional workflow model is discussed and a spectrum of related transaction models is covered in detail. A plethora of influential projects in workflow and process automation is summarized. The projects are drawn from both academia and industry. The monograph also provides a short overview of the most popular workflow management products, and the state of the workflow industry in general. Workflow and Process Automation: Concepts and Technology offers a road map through the shortcomings of existing solutions of process improvement by people with daily first-hand experience, and is suitable as a secondary text for graduate-level courses on workflow and process automation, and as a reference for practitioners in industry.

This well-organized textbook provides the design techniques of algorithms in a simple and straight forward manner. The book begins with a description of the fundamental concepts such as algorithm, functions and relations, vectors and matrices. Then it focuses on efficiency analysis of algorithms. In this unit, the technique of computing time complexity of the algorithm is discussed along with illustrative examples. Gradually, the text discusses various algorithmic strategies such as divide and conquer, dynamic programming, Greedy algorithm, backtracking and branch and bound. Finally the string matching algorithms and introduction to NP completeness is discussed. Each algorithmic strategy is explained in stepwise manner, followed by examples and pseudo code. Thus this book helps the reader to learn the analysis and design of algorithms in the most lucid way.

Boolean Algebra and Combinational NetworksPrinciple of Duality; Boolean Formulas and Functions : Normal Formulas; Canonical Formulas : Minterm Canonical Formulas, m-Notation; Manipulations of Boolean Formulas: Equation Complementation, Expansion about a Variable, Equation Simplification, The Reduction Theorems, Minterm Canonical Formulas, Maxterm Canonical Formulas, Complements of Canonical Formulas; Gates and Combinational Networks : Gates, Combinational Networks, Analysis Procedure, Synthesis Procedure, A Logic Design Example; Incomplete Boolean Functions and Don't Care Conditions : Describing Incomplete Boolean Functions, Don't Care Conditions in Logic Design; Additional Boolean Operations and Gates : The NAND-Functions, The NOR-Functions, Universal Gates, NAND-Gate Realizations, NOR-Gate Realizations, The Exclusive-OR-Function, The Exclusive-NOR Function.Simplification of Boolean ExpressionsFormulation of the Simplification Problem : Criteria of Minimality, The Simplification Problem; Prime Implicants and Irredundant Disjunctive Expressions : Implies, Subsumes, Implicants and Prime Implicants, Irredundant Disjunctive Normal Formulas; Prime Implicants and Irredundant Conjunctive Expressions; Karnaugh Maps : One-Variable and Two-Variable Maps, Three-Variable and Four-Variable Maps, Karnaugh Maps and Canonical Formulas, Product and Sum Term Representations on Karnaugh Maps; Using Karnaugh Maps to Obtain Minimal Expressions for Complete Boolean Functions : Prime Implicants and Karnaugh Maps, Essential Prime Implicants, Minimal Sums, Minimal Products;

Minimal Expressions of Incomplete Boolean Functions : Minimal Sums, Minimal Products; The Quine-McCluskey Method of Generating Prime Implicants and Prime Implicates : Prime Implicants and the Quine - McCluskey Method, Algorithm for Generating Prime Implicants, Prime Implicates and the Quine - McCluskey Method; Prime Implicant/Prime-Implicate Tables and Irredundant Expressions; Petrick's Method of Determining Irredunant Expressions, Prime-Implicate Tables and Irredundant Conjunctive Normal Formulas; Prime Implicant/Prime-Implicate Table Reductions : Essential Prime Implicants, Column and Row Reductions, A Prime - Implicant Selection Procedure; Decimal Method for Obtaining Prime Implicants; Map Entered Variables.Logic Levels and FamiliesLogic Levels, Integration Levels; Output Switching Times, The Propagation Delay, Fan-out and Fan-in, Extension to Other Logic Gates, Logic Cascades.Transistor-Transistor logic; Wired logic, TTL with Totem-Pole output, Thee-state output TTL, Schottky TTL; The MOS Field-Effect-Transistor : Operation of n-Channel, Enhancement-Type MOSFET, The n-Channel Depletion-Type MOSFET, The p-channel MOSFETs, Circuit Symbols, The MOSFET as a Resistor; NMOS and PMOS Logic : The NMOS Inverters, NMOS NOR-Gate, NMOS NAND-Gate, PMOS Logic, performance; The CMOS Inverter, CMOS NOR-Gate, CMOS NAND-Gate, performance, Comparison of the above logic families.Logic Design with MSI Components and Programmable Logic DevicesBinary Adders and Subtractors; Binary Subtractors, Carry Lookahead Adders; Decimal Adders; Comparators; Decoders; Logic Design Using Decoders; Decoders with an Enable Input; Encoders; Multiplexers; Logic Design with Multiplexers; Programmable Logic Devices (PLDs); PLD Notation; Programmable Read-Only Memories (PROMs); Programmable Logic Arrays (PLAs); Programmable Array Logic (PAL) Devices.Flip-Flops and Simple Flip-Flop ApplicationsThe Basic Bistable Element; Latches; The SR Latch, An Application of the SR Latch : A Switch Debouncer, The SR Latch, The Gated SR Latch, The Gated D Latch; Master-Slave Flip-Flops (Pulse-Triggered Flip-Flops); The Master-Slave SR Flip-Flop; The Master-Slave JK Flip-Flop; Edge-Triggered Flip-Flop; The Positive Edge-Triggered D Flip-Flop; Negative Edge-Triggered D flip-flops; Characteristic Equations; Registers; Counters : Binary Ripple Counters, Synchronous Binary Counters, Counters Based on Shift Registers ; Design of Synchronous Counters : Design of a Synchronous Mod-6 Counter Using Clocked JK Flip-Flops, Design of a Synchronous Mod-6 Counter Using Clocked D,T or SR Flip-Flops.Synchronous Sequential NetworksStructure and Operation of Clocked Synchronous Sequential Networks; Analysis of Clocked Synchronous Sequential Networks; Excitation and Output Expressions, Transition Equations, Transition Tables, Excitation Tables, State Tables, State Diagrams Network Terminal Behavior.

Taking a learn-by-doing approach, Software Engineering Design: Theory and Practice uses examples, review questions, chapter exercises, and case study assignments to provide students and practitioners with the understanding required to design complex software systems. Explaining the concepts that are immediately relevant to software designers, it begins with a review of software design fundamentals. The text presents a formal top-down design process that consists of several design activities with varied levels of detail, including the macro-, micro-, and construction-design levels. As part of the top-down approach, it provides in-depth coverage of applied architectural, creational, structural, and behavioral design patterns. For each design issue covered, it includes a step-by-step breakdown of the execution of the design solution, along with an evaluation, discussion, and justification for using that particular solution. The book outlines industry-proven software design practices for leading large-scale software design efforts, developing reusable and high-quality software systems, and producing technical and customer-driven design documentation. It also: Offers one-stop guidance for mastering the Software Design & Construction sections of the official Software Engineering Body of Knowledge (SWEBOK®) Details a collection of standards and guidelines for structuring high-quality code Describes techniques for analyzing and evaluating the quality of software designs Collectively, the text supplies comprehensive coverage of the software design concepts students will need to succeed as professional design leaders. The section on engineering leadership for software designers covers the necessary ethical and leadership skills required of software developers in the public domain. The section on creating software design documents (SDD) familiarizes students with the software design notations, structural descriptions, and behavioral models required for SDDs. Course notes, exercises with answers, online resources, and an instructor’s manual are available upon qualified course adoption. Instructors can contact the author about these resources via the author's website: http://softwareengineeringdesign.com/

This volume combines the proceedings of the 1987 SEI Conference on Software Engineering Education, held in Monroeville, Pennsylvania on April 30 and May 1, 1987, with the set of papers that formed the basis for that conference. The conference was sponsored by the Software Engineering Institute (SEI) of Carnegie-Mellon University. SEI is a federally-funded research and development center established by the United States Department of Defense to improve the state of software technology. The Education Division of SEI is charged with improving the state of software engineering education. This is the third volume on software engineering education to be pub lished by Springer-Verlag. The first (Software Engineering Education: Needs and Objectives, edited by Tony Wasserman and Peter Freeman) was published in 1976. That volume documented a workshop in which educa tors and industrialists explored needs and objectives in software engineering education. The second volume (Software Engineering Education: The Educational Needs of the Software Community, edited by Norm Gibbs and Richard Fairley) was published in 1986. The 1986 volume contained the proceedings of a limited attendance workshop held at SEI and sponsored by SEI and Wang Institute. In contrast to the 1986 Workshop, which was limited in attendance to 35 participants, the 1987 Conference attracted approximately 180 participants.

Practical Guidance on the Efficient Development of High-Quality Software Introduction to Software Engineering, Second Edition equips students with the fundamentals to prepare them for satisfying careers as software engineers regardless of future changes in the field, even if the changes are unpredictable or disruptive in nature. Retaining the same organization as its predecessor, this second edition adds considerable material on open source and agile development models. The text helps students understand software development techniques and processes at a reasonably sophisticated level.

Students acquire practical experience through team software projects. Throughout much of the book, a relatively large project is used to teach about the requirements, design, and coding of software. In addition, a continuing case study of an agile software development project offers a complete picture of how a successful agile project can work. The book covers each major phase of the software development life cycle, from developing software requirements to software maintenance. It also discusses project management and explains how to read software engineering literature. Three appendices describe software patents, command-line arguments, and flowcharts.

Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress's highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words "at work" suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names. Having digested everyone's feedback, we selected 15 folks who've been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow Joe Armstrong: Inventor of Erlang Joshua Bloch: Author of the Java collections framework, now at Google Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger Douglas Crockford: JSON founder, JavaScript architect at Yahoo! L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1 Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal Dan Ingalls: Smalltalk implementor and designer Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler Donald Knuth: Author of The Art of Computer Programming and creator of TeX Peter Norvig: Director of Research at Google and author of the standard text on AI Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress Ken Thompson: Inventor of UNIX Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker

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.

Turbo machines, in mechanical engineering, describes machines that transfer energy between rotor and fluid, including turbines, pumps and compressors. While turbine transfers energy from fluid to rotor and compressor and a pump transfers energy from rotor to fluid. Turbo machine is a power or a head generating machine which employs the dynamic action of a rotating element, the rotor; the action of the rotor changes the energy level of the continuously flowing fluid through the machine. The majority of turbo machines run at comparatively higher speeds without any mechanical problems and high volumetric efficiency. Turbo machines can be categorised on the basis of the nature of flow path through the passage of the rotor. The same fundamentals are applicable to all turbo machines, certainly there are significant differences between these machines. In this book SI unit system is followed. Our hope is that this book, through its careful explanations of concepts, practical examples and figures bridges the gap between knowledge and proper application of that knowledge.

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

Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering includes a set of rigorously reviewed world-class manuscripts addressing and detailing state-of-the-art research projects in the areas of Computer Science, Software Engineering, Computer Engineering, and Systems Engineering and Sciences. Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering includes selected papers form the conference proceedings of the International Conference on Systems, Computing

Sciences and Software Engineering (SCSS 2007) which was part of the International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE 2007).

Computer Systems Engineering Management provides a superb guide to the overall effort of computer systems bridge building. It explains what to do before you get to the river, how to organise your work force, how to manage the construction, and what do when you finally reach the opposite shore. It delineates practical approaches to real-world development issues and problems presents many examples and case histories and explains techniques that apply to everything from microprocessors to mainframes and from person computer applications to extremely sophisticated systems

Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows users exactly how to spot the best opportunities for refactoring and exactly how to do it, step by step.

A "good" programmer can outproduce five, ten, and sometimes more run-of-the-mill programmers. The secret to success for any software company then is to hire the good programmers. But how to do that? In Joel on Hiring, Joel Spolsky draws from his experience both at Microsoft and running his own successful software company based in New York City. He writes humorously, but seriously about his methods for sorting resumes, for finding great candidates, and for interviewing, in person and by phone. Joel’s methods are not complex, but they do get to the heart of the matter: how to recognize a great developer when you see one.

“This is an incredibly wise and useful book. The authors have considerable real-world experience in delivering quality systems that matter, and their expertise shines through in these pages. Here you will learn what technical debt is, what is it not, how to manage it, and how to pay it down in responsible ways. This is a book I wish I had when I was just beginning my career. The authors present a myriad of case studies, born from years of experience, and offer a multitude of actionable insights for how to apply it to your project.” –Grady Booch, IBM Fellow Master Best Practices for Managing Technical Debt to Promote Software Quality and Productivity As software systems mature, earlier design or code decisions made in the context of budget or schedule constraints increasingly impede evolution and innovation. This phenomenon is called technical debt, and practical solutions exist. In Managing Technical Debt, three leading experts introduce integrated, empirically developed principles and practices that any software professional can use to gain control of technical debt in any software system. Using real-life examples, the authors explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They introduce proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and “paying off” debt over time. They describe how to establish managing technical debt as a core software engineering practice in your organization. Discover how technical debt damages manageability, quality, productivity, and morale–and what you can do about it Clarify root causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure Identify technical debt items, and analyze their costs so you can prioritize action Choose the right solution for each technical debt item: eliminate, reduce, or mitigate Integrate software engineering practices that minimize new debt Managing Technical Debt will be a valuable resource for every software professional who wants to accelerate innovation in existing systems, or build new systems that will be easier to maintain and evolve.

ASD S1000D is an internationally recognized and utilized standard for creating technical data. A common source database is used to contain all of the files that make up a technical publication, and all content is modular. Managing an S1000D project well requires a lot of up-front planning and preparation. There are so many considerations that taking on such a project can be quite overwhelming. This book, Managing Your First S1000D Project, is a guide to help you, particularly through the most difficult part of an S1000D project: Set-up. The second edition contains elaboration on important concepts, more focus on the most current Issues of S1000D, a better chapter structure, and more illustrations of important content.