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. It applies to the entire development process of embedded software, it can be used for project management, development, testing and quality assurance of embedded software. It is also useful for technical writers, developer advocates, product managers, and other technical roles that create and contribute to documentation for their products and services.

The book offers several different ways to interact and strives to create a truly collaborative environment where new ideas and exciting findings can be presented and shared. For example, this year’s open space session, which was “a conference within a conference”, was larger than ever before. Agile software development is a unique phenomenon from several perspectives. This volume contains papers presented at the International Conference on Software Process (ICSP) held in Vancouver, Canada, during May 16-17, 2009. ICSP 2009 was the third conference of the ICSP series, continuing the software process workshops begun 25 years ago. The theme of ICSP 2009 was “Processes to Develop Trustworthy Software.” Software development takes place in a dynamic context of constantly changing technologies and limited resources. Teams worldwide are under increasing pressure to deliver trustworthy software products more quickly and with higher levels of quality. At the same time, global competition is forcing software development organizations to cut costs by rationalizing processes, outsourcing parts of all of their activities, re-using existing software in new or modified applications and evolving existing systems to meet new needs, while still minimizing the risk of projects failing to deliver. To address these difficulties, new or modified processes are emerging including lean and agile methods, plan-based product line development, and increased integration with systems engineering processes. Papers present research and real-world experiences in many areas of software and system processes, including software measurement, software process improvement, software project management, software quality, software architecture, software configuration control, software reuse, software maintenance, software project risk management, software process improvement, software fault detection and recovery, software quality assurance, and software process evaluation.

Software development and information systems projects have always been subject to continuous scrutiny and evaluation. Many, however, critical software development is vital for the success of an information system. Software Development Techniques for Constructive Information Systems Design focuses on the aspects of information systems and software development as a merging process. This reference source pays special attention to the emerging research, trends, and experiences in this area which is bound to enhance the reader’s understanding of the growing and ever-evolving field. Academics, researchers, students, and working professionals in this field will benefit from this publication’s unique perspective.

This book presents the thoroughly revised and refereed proceedings of the 15th Monterey Workshop, held in Budapest, Hungary, September 24-26, 2009. The theme of the workshop was Foundations of Computer Software, Future Trends and Techniques for Development. The 13 revised full papers presented at the workshop explore, how the foundations and development techniques of computer software could be adapted to address such a challenge. Material presented in the papers spans the whole software life cycle, starting from specification and analysis, design and the choice of architecture, large scale, real-world software development, code generation and configuration, deployment, and evolution.

This second volume on software engineering processes includes reprinted and newly authored papers that describe the supporting life cycle processes in a manner that can prepare individuals to take the IEEE Computer Society Certified Professional Software Developer (CPSD) examination.

Advances in Systems, Computing Sciences and Software Engineering This book includes the proceedings of the International Conference on Systems, Computing Sciences and Software Engineering (ICSCSE). The proceedings are a set of rigorously reviewed world-class manuscripts and proceedings and addressing state-of-the-art research areas in the fields of computer science, software engineering, engineering, systems sciences and engineering, information technology, parallel and distributed computing and web-based programming. ICSCSE was part of the International Joint Conferences on Computer, Informatics, and Systems Sciences and Engineering (ICCCISE) (www.isccise.org; the World's First Engineering/Computing and Systems Research E-Conference. ICSCSE was the first high-caliber Research Conference in the world to be completely conducted online in real-time via the internet. ICSCSE received 255 research paper submissions and the final program included 145 accepted papers, from more than 45 countries. The concept and format of ICSCSE were very exciting and ground-breaking. The Powerpoint presentations, final paper manuscripts and time schedule for live presentations over the web had been available for 3 weeks prior to the conference. The proceedings also represented live audio and video streaming from the permanent ICSCSE archive, which also included all power point presentations and papers. SCSE provided a virtual forum for presentation for discussion of the state-of-the-art research on Systems, Computing Sciences and Software Engineering.

This revised edition of Software Engineering-Principles and Practices has become more comprehensive with the inclusion of several topics. The book now offers a complete understanding of software engineering as an engineering discipline. It provides the reader with an in-depth coverage of software development and engineering. It is a fundamental book for anyone involved with software development. The book also includes a comprehensive overview of the software engineering discipline, including the principles, methods, tools, and techniques of software engineering. The book is intended for anyone involved with software development and engineering, including software engineers, managers, and researchers.

Learn to integrate programming with good documentation. This book teaches you the craft of documentation for each step in the software development lifecycle, from understanding your users’ needs to publishing, measuring, and maintaining useful documentation. Hence, our end-users get documented software, which after adequate documentation suffer from poor developer productivity, project scalability, user adoption, and accessibility. In short: bad documentation kills projects. Docs for Developers promises the process of creating great developer documentation, following a toolkit for software developers as they work to launch a new product. At each step along the way, the book shows you how to create, build, distribute, and maintain your documentation. You’ll learn how to create documentation, manage version control, and integrate your users’ frustrations. Saunder’s best practices and guidelines in this book will help you create documentation that will keep your users happy and engaged with your product.

(products, from: sales@chinesestandard.net) This standard specifies the quality assurance process and general requirements for the development of embedded software projects. This standard applies to the entire development process of embedded software, it can be used for project management, development, testing and quality assurance of embedded software. This standard applies to all embedded software development projects. It does not apply to the development of non-embedded software or to the development of systems that are not embedded.
engineering. How can software engineers manage a codebase that evolves as requirements change and demands over the length of its life? Based on their experience at Google, software engineers Titus Winter and Byron Knight, along with technical writer Tom Mahnken, 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 affect the management of an engineering organization, especially in the area of software development, architecting, writing, and maintaining code. How time affects the sustainability of software and how you can make your code resilient over time. How scale affects the viability of an engineering organization. What trade-offs a typical engineer needs to make when evaluating design and development decisions.

Literate programming is a programming methodology that combines a language with a documentation language, making programs more easily maintained than programs written only in a high-level language. A literate programmer is an essayist who writes documentation as part of the program. This anthology of essays includes Knuth's early papers on related topics such as structured programming as well as the Computer Journal article that launched literate programming. Many examples are given, including excerpts from the programs for TeX and METAFONT. The final essay is an essay on CWEB, a literate programming system in C and related languages. How software process definition, documentation, and improvement should be an integral part of every software engineering organization. This book addresses the specific documentation requirements in support of the CMMI 3BM by providing detailed documentation guidelines in the form of Detailed organizational policy example templates. An integrated set of over 20 deployable document templates. Examples of over 50 common work products required in support of assessment activities. Examples of organizational delineation of software process. This book provides a complete set of IEEE Software Engineering Standards-based template examples that support the documentation required for all activities associated with software development projects. The goal is to provide practical support for individuals responsible for the development and documentation of software processes and procedures. The objective is to present the reader with an integrated set of documents that support the requirements of the CMMI-3BM Levels 2 and 3. This book is meant to both complement and extend the information provided in Jumpstart CMMI/CMM® Software Process Improvement Using IEEE Software Engineering Standards. Jumpstart provides a detailed mapping of both the CMMI and the ISO standards to the IEEE standards and contains specific details about the most common types of software organizations. The book builds on the latest thinking in software engineering, project management, and quality assurance for software development. The tools NSE CLICK—an automatic acceptance testing platform for outsourcing (or internally developed) C/C++ products, and NSE CLICK_J—an automatic acceptance testing platform for outsourcing (or internally developed) Java products are particularly designed for non-technical readers to view/review how the acceptance testing of a software product developed with NSE can be performed automatically, and how the product developed with NSE is truly maintainable at the customer site.

Part of the new Alyn & Bacon series in technical communication, Writing Software Documentation features a step-by-step strategy to writing and describing procedures. This task-oriented book is designed to support both college students taking a course and professionals working in the field. Teaching apparatus includes complete programs for students to work on and a full set of project tracking forms, as well as a broad range of examples including Window-style pages and screen shots and award-winning examples from ICT competitions.

Use an Approach Inspired by Domain-Driven Design to Build Documentation That Evolves to Maximize Value Throughout Your Development Software documentation can come to life, stay dynamic, and actually help you build better software. Writing for the people who use your software, and others who need to understand how your software works, requires a new mindset. Here, Cyril Callon explores how to use non-prescriptive, language to help your software become more than just a collection of functions. He lays the groundwork for building the kinds of software that are easy to maintain and extend, and he explains how to use the software you write to make it easier to build software.

This book brings together experts to discuss relevant results in software process modeling, and expresses their personal view of this field. It is designed for a professional audience of researchers and practitioners in industry, and graduate-level students.

Overview and Goals The agile approach for software development has been applied more and more extensively since the mid nineties of the 20th century. Though there are only about ten years of accumulated experience using the agile approach, it is currently conceived as one of the mainstream approaches for software development. This book presents a complete software engineering course from the agile angle. Our intention is to present the agile approach in a holistic and comprehensible way, with emphasis on learning and interprocess cooperation between teams, customers, and management. The Organizational perspective, which includes managerial and cultural aspects, and refers to software project management and context of the software engineering process. The Technical perspective, which includes operational and technical aspects, and refers to software engineering process. The book is aimed at software engineers, project managers, and software engineering managers.

This book presents a comprehensive discussion on software quality issues and software quality assurance (SQA) principles and practices, and lays special emphasis on implementing and managing SQA. Primarily designed to serve three audiences: book learning and college students, vocational training participants, and software engines and software development managers, the book may be applicable to all personnel engaged in a software projects. A wide view of SQA. The book delves into SQA issues, going beyond the classic boundaries of custom-made software development to also cover in-house software development, subcontractors, and readymade software. An up-to-date wide-range coverage of SQA and SQA related topics. Providing comprehensive coverage on multifarious SQA subjects, including topics, hardly explored till in SQA texts. A systematic presentation of the SQA Function and its tasks: establishing the SQA processes, planning, coordinating, follow-up, review and evaluation of SQA processes. Focus on SQA implementation issues. Specialized chapter sections, examples, implementation tips, and topics for discussion. Pedagogical support: Each chapter includes a case study, examples, a summary, selected bibliography, review questions and tasks for discussion. The book is also supported by an Instructor’s Guide.

Software engineering requires specialized knowledge of a broad spectrum of topics, including the construction of software and the platform, 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 E. Laplacote 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 edition for those who use software engineering students, IT professionals, researchers, managers, and scholars with uncritical coverage of the topic that encompasses 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 export options. Contact Taylor and Francis for more information or to inquire about subscription options and print/online combination packages. OS: (Tel) 1.888.316.2376 (E-mail) e-reforders@taylorandfrancis.com International (Tel) +44 (0) 20 7107 6002; (E-mail) online.sales@tandf.co.uk
Computer systems play an important role in our society. Software drives these systems. Massively investments of time and resources are made in developing and implementing these systems. Maintenance is inevitable. It is hard and costly. Considerable resources are required to keep the systems active and dependable. We cannot maintain software unless maintainability characters are built into the products and processes. There is an urgent need to reinforce software development practices based on quality and reliability principles. Though maintenance is a mini development lifecycle, it has its own problems. Maintenance issues need corresponding tools and techniques to address them. Software professionals are key players in maintenance. While development is an art and science, maintenance is a craft. We need to develop maintenance personnel to master this craft. Technology impact is very high in systems world today. We can no longer conduct business in the way we did before. That calls for reengineering system and software. Even reengineered software needs maintenance, soon after its implementation. We have to take business knowledge, procedures, and data into the newly reengineered world. Software maintenance people can play an important role in improving the performance, marketing, and eventually the efficiency in their way of working. Systems engineers are dealing with changing software requirements, changing databases service into corporate data warehouses. Software engineering environments, rapid application development tools are changing the way we used to develop and maintain software. Software maintenance is moving from code maintenance to data and system maintenance. Software engineers need to work with the business to understand the evolution and evolutionary characteristics of software systems. Software professionals have to maintain not only the software, but the momentum of change in systems and software. In this study, we observe various issues, tools and techniques, and the ways to overcome them. We also try to find the role of technology with the way to manage them, live with them, and control their negative impact. 

Looking for a way to invigorate your technical writing team and grow that expertise to include developers, designers, and writers of all backgrounds? When you treat docs like code, you multiply everyone's efforts and streamline processes through collaboration, automation, and innovation. Second edition now available with updates and more information about version control for documents and continuous publishing.

The book describes how to manage and successfully deliver large, complex, and expensive systems that can be composed of millions of lines 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.

A vital new publication for scientists and researchers in the field, this book constitutes the refereed proceedings of the 8th International Conference on Product Focused Software Process Improvement, PROFESS 2007, held in Riga, Latvia in July 2007. The 29 revised full papers, along with four short papers, and four keynote addresses accepted for PROFESS 2007 were carefully reviewed and selected from 55 submissions. The papers constitute a balanced mix of academic and industrial aspects they cover a range of topics for ease of access.

This book contains the refereed proceedings of the 15th International Conference on Agile Software Development, XP 2014, held in Rome, Italy, in May 2014. Because of the wide application of agile approaches in industry, the need for collaboration between academics and practitioners has increased in order to develop the body of knowledge available to support managers, system engineers, and software engineers in their managerial/economic and architectural/project/technical decisions. Year after year, the XP conference has facilitated such improvements and provided evidence on the advantages of agile methodologies by examining the latest theories, practical applications, and implications of agile and lean methods. The 15 full papers, seven short papers, and four experience reports accepted for XP 2014 were selected from 55 submissions and are organized in sections on: agile development, agile challenges and contracting, lessons learned and agile maturity, how to evolve software engineering teaching, methods and metrics, and lean development.

Software engineering research has different profiles in Europe and North America. North Americans isolate knowledge into the practical, technical, and organizational aspects of software engineering. In Europe, the work concentrates more on foundations, and formal modeling of software engineering issues. Appropriate software architecture models can be a basis for the application of software architectures in practice. Research-driven software practice is fast growing in software engineering roles in the danger of developing into a niche held fading to 100 a solid scientific basis or to contribute substantially to the progress in software engineering. Work concentrating on formal aspects alone is in the danger of becoming too technical and isolated from practice so that any transfer into practical application will fail. Substantial progress in software engineering can be achieved, however, by bringing together pragmatic and foundational work in software engineering - search. This can provide a step towards a common scientific basis for software engineering that allows us to integrate the various research results to fruitful synergistic effects. It will also help to identify critical research paths and to develop an adequate paradigm for the scientific disciplines of software engineering. In software and systems engineering it is necessary to reconcile the paradigm driven and software management driven approaches. The problem of the software development problems associated with the old techniques are symptoms of a lack of formalization and a lack of automation support. It was the goal of this workshop to bring together experts from science and practice in software systems engineering from North America and Europe.

Freely available source code, with contributions from thousands of programmers worldwide, is a central component of the success of the open source software movement. Underlying this revolution is a new way of developing software, where the power of the open-source development model, OL, is becoming a viable mainstream alternative to commercial software systems. How Open Sources, leaders of Open Sources, come together for the first time to discuss the new vision of the software industry and its consequences. The essays in this volume offer insight into how the Open Source movement works, why it succeeded, and where it is going. For programmers who have labored on open-source projects, Open Sources is the new gospel: a powerful vision from the movement's spiritual leaders. For businesses integrating open-source software into their enterprise, Open Sources reveals the mysteries of how open development builds better software, and how businesses can leverage freely available software for a competitive business advantage. The contributors here have been the leaders in the open-source arena Brian Behlendorf (Apache); Richard Stallman (GNU, Free Software Foundation, Emacs) Michael Tiemann (Cygnus Solutions) Linus Torvalds (Linux) Paul Vixie (BSD) Larry Wall (Perl) This book explains why the major of the Internet's servers use open-source technologies for everything from the operating system to Web serving and email. How open-source technology has overtaken and surpassed the commercial efforts of billion dollar companies like Microsoft and IBM to dominate software markets. Learn the inside story of what led Netscape to release its source code under the open-source open license, why open-source technology continues to grow, and why open-source technology will develop even further.

The book describes how to manage and successfully deliver large, complex, and expensive systems that can be composed of millions of lines 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.
in an application-oriented way. It answers questions such as: * How do we analyze an application domain utilizing the knowledge and experience of the users? * What is the proper software architecture for large, distributed interactive systems that can utilize UML and design patterns? * Where and how should we utilize the techniques and methods of the Unified Process and Xtreme Programming? This book brings together the best of research, development, and day-to-day project work. *The strength of the book is that it focuses on the transition from design to implementation in addition to its overall vision about software development.* -Bent Bruun Kristensen, University of Southern Denmark, Denmark

A structured approach to systems development that views the creation of documentation as a combined effort of systems planners, analysts, programmers, and technical writers. Includes alternatives for each of the documents presented and a chapter on integrating a systems development methodology with CASE. Annotation copyrighted by Book News, Inc., Portland, OR

This book contains the refereed proceedings of the 16th International Conference on Agile Software Development, XP 2015, held in Helsinki, Finland, in May 2015. While agile development has already become mainstream in industry, this field is still constantly evolving and continues to spur an enormous interest both in industry and academia. The XP conference series has always played, and continues to play, an important role in connecting the academic and practitioner communities, providing a forum for both formal and informal sharing and development of ideas, experiences, and opinions. The theme of XP 2015 “Delivering Value: Moving from Cyclic to Continuous Value Delivery” reflects the modern trend towards organizations that are simultaneously very efficient and flexible in software development and delivery. The 15 full and 7 short papers accepted for XP 2015 were selected from 44 submissions. All of the submitted papers went through a rigorous peer-review process. Additionally, 11 experience reports were selected from 45 proposals, and in each case the authors were shepherded by an experienced researcher.

Software Engineering for Embedded Systems: Methods, Practical Techniques, and Applications, Second Edition provides the techniques and technologies in software engineering to optimally design and implement an embedded system. Written by experts with a solution focus, this encyclopedic reference gives an indispensable aid on how to tackle the day-to-day problems encountered when using software engineering methods to develop embedded systems. New sections cover peripheral programming, Internet of things, security and cryptography, networked and packet processing, and hands on labs. Users will learn about the principles of good architecture for an embedded system, design practices, details on principles, and much more. Provides a roadmap of key problems/issues and references to their solution in the text. Reviews core methods and how to apply them. Contains examples that demonstrate timeless implementation details. Users can study to show how key ideas can be implemented, the rationale for choices made, and design guidelines and trade-offs

Software Engineering discusses the major issues associated with different phases of software development life cycle. Starting from the basics, the book discusses several advanced topics. Topics like software project management, software process models, developing methodologies, software specification, software testing and quality, software implementation, software security, software maintenance and software reuse are discussed. This book also gives an introduction to the new emerging technologies, trends and practices in software engineering field. New topics such as MIMO technology, AJAX, etc. are included in the book. The topics like .NET framework, J2EE, etc. are also dealt with. Case studies, discussions on real-life situations of dealing with IT related problems and finding their solutions in an easy manner, are given in each chapter. Elegant and simple style of presentation makes the reading of this book a pleasant experience. Students of Computer Science and Engineering, Information Technology and Computer Applications should find this book highly useful. It would also be useful for IT technology professionals who are interested to get acquainted with the latest and the newest technologies.

This book contains the refereed proceedings of the 13th International Conference on Agile Software Development, XP 2012, held in Malmö, Sweden, in May 2012. In the last decade, we have seen agile and lean software development strongly influence the way software is developed. Agile and lean software development has moved from being a way of working for a number of pioneers to becoming more or less, the expected way of developing software in industry. The topics covered by the selected full papers include general aspects of agility, agile teams, studies related to the release and maintenance of software, and research on specific practices in agile and lean software development. They are complemented by four short papers capturing additional aspects of agile and lean projects.

The complexity of software is continuously growing as a result of today’s interconnected business processes. Governance of architecture and technology strategy helps to ensure coherence of software and avoid excessive complexity. At the same time software development needs room for creativity and empowerment to provide solutions to business problems of increasing complexity. The book looks at this software dilemma from the perspectives of CIOs/CTOs, software architects, and auditors. Each of these groups has different interests which need to be considered, reconciled, and balanced. CIOs/CTOs are provided with the boundary conditions they have to establish assuring the achievement of strategic objectives. Architects and auditors find proven concepts for effectively assessing software projects and architectures, as well as for effectively communicating identified issues to responsible persons. The book is based on the author’s long experience in software engineering, governance, and auditing.

This book constitutes the refereed proceedings of the 7th International Conference on Extreme Programming and Agile Processes in Software Engineering, XP 2006, held in Oulu, Finland, June 2004. The book presents 16 revised full papers together with 6 experience papers, 12 poster papers and panel summaries, organized in topical sections on foundation and rationale for agile methods, effects of pair programming, quality in agile software development, and more.