

Estimating Software Intensive Systems Projects Products And Processes Paperback Sei Series In Software Engineering

Evolution of software has long been recognized as one of the most problematic and challenging areas in the field of software engineering, as evidenced by the high, often up to 60-80%, life-cycle costs attributed to this activity over the life of a software system. Studies of software evolution are central to the understanding and practice of software development. Yet it has received relatively little attention in the field of software engineering. This book focuses on topics aimed at giving a scientific insight into the aspect of software evolution and feedback. In summary, the book covers conceptual, phenomenological, empirical, technological and theoretical aspects of the field of software evolution - with contributions from the leading experts. This book delivers an up-to-date scientific understanding of what software evolution is, to show why it is inevitable for real world applications, and it demonstrates the role of feedback in software development and maintenance. The book also addresses some of the phenomenological and technological underpinnings and includes rules and guidelines for increased software evolvability and, in general, sustainability of the evolution process. Software Evolution and Feedback provides a long overdue, scientific focus on software evolution and the role of feedback in the software process, making this the indispensable guide for all software practitioners, researchers and managers in the software industry.

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

This research entailed calibration and validation of the SAGE software Cost/Schedule Estimating System, Version 1.7 as a means to improve estimating accuracy for DoD software-intensive systems, and thereby introduce stability into software system development. SAGE calibration consisted of using historical data from completed projects at the Space and Missile Systems Center (SMC) and the Electronic Systems Center (ESC) to derive average performance factors (i.e., calibration factors) for pre-defined categories of projects. A project was categorized for calibration by either its primary application or by the contractor that developed it. The intent was to determine the more appropriate categorization for calibration. SAGE validation consisted of using the derived calibration factors to predict completed efforts, not used in deriving the factors. Statistical resampling employing Monte Carlo simulation was used to calibrate and validate the model on each possible combination of a category's projects. Three statistical measures were employed to measure model performance in default and calibrated estimating modes. SAGE generally did not meet pre-established criteria for estimating accuracy, although the model demonstrated some improvement with calibration. Calibration of projects categorized by contractor resulted in better calibrated model performance than calibration of projects categorized by application. This categorization is suggested for future consideration.

Software legend Capers Jones reveals the tight links between software quality, ROI, and TCO, and help you optimize all three • •Strong empirical evidence that high quality generates strongly positive ROI and reduced TCO. •Practical ways to prevent defects, and remove them in pre-test, test, and postrelease. •Easy checklists for assessing and improving practice, plus insights into the costs/benefits of intervention. •By renowned software consultant Capers Jones. In this book, world-renowned software management expert Capers Jones and software quality guru Jitendra Subramanyam help development leaders and practitioners quantify and optimize the economic impact of quality throughout the software lifecycle - and then choose the highest value interventions to improve it. The authors introduce powerful empirical and field data on the ability of inspection, static analysis, and test methods to reduce up to 95% of defects, and discuss the business value of improvements of this magnitude. The Economics of Software Quality is based on proven best quality practices in IT departments and at world-leading integrators, embedded software companies, and systems software groups. Jones and Curtis bring together crucial new information on: • •Identifying and fixing the root causes of short- and long-term software cost inefficiencies. •Predicting and measuring software defects and their quality impacts. •Assessing current practices and identifying the best interventions. •Calculating the ROI of quality during development and maintenance. •Comparing and choosing methods of defect prevention. •Selecting methods of defect removal, such as inspections and static analysis. •Understanding and evaluating more than 20 kinds of software testing. •Best practices for postrelease defect reporting and repair. •Recognizing 'hazardous' metrics and their problems

"This book addresses the complex issues associated with software engineering environment capabilities for designing real-time embedded software systems"--Provided by publisher.

Architectural design is a crucial first step in developing complex software intensive systems. Early design decisions establish the structures necessary for achieving broad systemic properties. However, today's organizations lack synergy between software their development processes and technological methodologies. Providing a thorough treatment of

A guide to the application of the theory and practice of computing to develop and maintain software that economically solves real-world problem How to Engineer Software is a practical, how-to guide that explores the concepts and techniques of model-based software engineering using the Unified Modeling Language. The author—a noted expert on the topic—demonstrates how software can be developed and maintained under a true engineering discipline. He describes the relevant software engineering practices that are grounded in Computer Science and Discrete Mathematics. Model-based software engineering uses semantic modeling to reveal as many precise requirements as possible. This approach separates business complexities from technology complexities, and gives developers the most freedom in finding optimal designs and code. The book promotes development scalability through domain partitioning and subdomain partitioning. It also explores software documentation that specifically and intentionally adds value for development and maintenance. This important book: Contains many illustrative examples of model-based software engineering, from semantic model all the way to executable code Explains how to derive verification (acceptance) test cases from a semantic model Describes project estimation, along with alternative software development and maintenance processes Shows how to develop and maintain cost-effective software that solves real-world problems Written for graduate and undergraduate students in software engineering and professionals in the field, How to Engineer Software offers an introduction to applying the theory of computing with practice and judgment in order to economically develop and maintain software.

Reliability of Software Intensive Systems

Many software projects fail because their leaders don't know how to estimate, schedule, or measure them accurately. Fortunately, proven tools and techniques exist for every facet of software estimation. Estimating Software-Intensive Systems brings them together in a real-world guidebook that will help software managers, engineers, and customers immediately improve their estimates—and drive continuing improvements over time. Dick Stutzke presents here a disciplined and repeatable process that can produce accurate and complete estimates for any project, product, or process, no matter how new or unusual. Stutzke doesn't just describe formal techniques: He offers simple, easy-to-use templates, spreadsheets, and tools you can start using today to identify and estimate product size, performance, and quality—as well as project cost, schedule, and risk reserves. Stutzke shows how to quickly "get your arms around" users' problems and requirements, the structure of a solution, and the process needed to deliver it. You'll learn how to choose the most appropriate estimating techniques and tools; collect accurate data, track progress, and update estimates; and recalibrate estimating models to improve estimation accuracy. Stutzke's techniques apply whether you're creating custom in-house business software, purchasing or customizing "off-the-shelf" technology, or constructing complex, one-of-a-kind military, industrial, or commercial systems. These techniques apply to small and large projects, and to all project life cycles—from agile to plan-driven. This book will help you plan, estimate, budget, schedule, purchase, design, build, test, deploy, operate, and maintain software-intensive systems. It explains how to size software, identify all cost components, calculate the associated costs, and set a competitive price. A separate section covers topics of interest for large projects: designing an appropriate work breakdown structure, collecting data from cost accounting systems, and using earned value measurement. You'll find updates and even more information on this book's companion web site, <http://www.sw-estimation.com>.

Deliver bug-free software projects on schedule and within budget Get a clear, complete understanding of how to estimate software costs, schedules, and quality using the real-world information contained in this comprehensive volume. Find out how to choose the correct hardware and software tools, develop an appraisal strategy, deploy tests and prototypes, and produce accurate software cost estimates. Plus, you'll get full coverage of cutting-edge estimating approaches using Java, object-oriented methods, and reusable components. Plan for and execute project-, phase-, and activity-level cost estimations Estimate regression, component, integration, and stress tests Compensate for inaccuracies in data collection, calculation, and analysis Assess software deliverables and data complexity Test design principles and operational characteristics using software prototyping Handle configuration change, research, quality control, and documentation costs "Capers Jones' work offers a unique contribution to the understanding of the economics of software production. It provides deep insights into why our advances in computing are not matched with corresponding improvements in the software that drives it. This book is absolutely required reading for an understanding of the limitations of our technological advances." --Paul A. Strassmann, former CIO of Xerox, the Department of Defense, and NASA

An authoritative primer on managing software-based development projects and complex software/hardware systems Managing the Development of Software-Intensive Systems discusses the application of project management and general management techniques to large software development projects and complex software/hardware systems. Drawing upon the author's experience in developing a project management workshop for AT&T employees, as well as in teaching software engineering courses at Monmouth University and workshops for a variety of other audiences, this practical guide allows readers to reliably develop large software applications and systems that require the simultaneous development of electronic hardware and the software that controls the hardware. Integrates the project management processes of planning, organizing, monitoring, and control with the underlying technical processes used for product development Teaches how to plan and manage verification and validation for large software projects or complex software/hardware systems Explains what additional management activities must take place in organizations with a multi-project environment Discusses how inspection results and testing metrics can be used to monitor project status Describes techniques to help manage inherent risks in software-based product development Each chapter is accompanied by a case study based on an actual situation with which the author is familiar; this gives the reader experience in doing the management work. The author teaches readers how to use their own experience to improve the way they manage projects and provides a method for reviewing successes and failures to help increase their capabilities in the future. Managing the Development of Software-Intensive Systems serves as both an introduction to project management for software and hardware developers and as an

advanced material resource for experienced managers. The contents will benefit managers of software-based development projects and organizations, as well as organizations that outsource development work. This book can also be used as a textbook in undergraduate or graduate courses in computer engineering, computer science, software engineering, information technology, commerce, and administration with an information systems orientation.

"This book provides a detailed analysis of the important strategies for integrating IT systems into fields such as e-business and customer-relationship management. It supplies readers with a comprehensive survey of existing enterprise architecture and integration approaches, and presents case studies that illustrate best practices, describing innovative methods, tools, and architectures with which organizations can systematically achieve enterprise integration"--Provided by publisher.

Innovative tools and techniques for the development and design of software systems are essential to the problem solving and planning of software solutions. *Software Design and Development: Concepts, Methodologies, Tools, and Applications* brings together the best practices of theory and implementation in the development of software systems. This reference source is essential for researchers, engineers, practitioners, and scholars seeking the latest knowledge on the techniques, applications, and methodologies for the design and development of software systems.

Software effort estimation is one of the oldest and most important problems in software project management, and thus today there are a large number of models, each with its own unique strengths and weaknesses in general, and even more importantly, in relation to the environment and context in which it is to be applied. Trendowicz and Jeffery present a comprehensive look at the principles of software effort estimation and support software practitioners in systematically selecting and applying the most suitable effort estimation approach. Their book not only presents what approach to take and how to apply and improve it, but also explains why certain approaches should be used in specific project situations. Moreover, it explains popular estimation methods, summarizes estimation best-practices, and provides guidelines for continuously improving estimation capability. Additionally, the book offers invaluable insights into project management in general, discussing issues including project trade-offs, risk assessment, and organizational learning. Overall, the authors deliver an essential reference work for software practitioners responsible for software effort estimation and planning in their daily work and who want to improve their estimation skills. At the same time, for lecturers and students the book can serve as the basis of a course in software processes, software estimation, or project management.

The book describes how to manage and successfully deliver large, complex, and expensive systems that can be composed of millions of line of software code, being developed by numerous groups throughout the globe, that interface with many hardware items being developed by geographically dispersed companies, where the system also includes people, policies, constraints, regulations, and a myriad of other factors. It focuses on how to seamlessly integrate systems, satisfy the customer's requirements, and deliver within the budget and on time. The guide is essentially a "shopping list" of all the activities that could be conducted with tailoring guidelines to meet the needs of each project. The widespread deployment of millions of current and emerging software applications has placed software economic studies among the most critical of any form of business analysis. Unfortunately, a lack of an integrated suite of metrics makes software economic analysis extremely difficult. The International Function Point Users Group (IFPUG), a nonpro

PLEASE PROVIDE COURSE INFORMATION PLEASE PROVIDE

Critical Code contemplates Department of Defense (DoD) needs and priorities for software research and suggests a research agenda and related actions. Building on two prior books- *Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale* and *Preliminary Observations on DoD Software Research Needs and Priorities*-the present volume assesses the nature of the national investment in software research and, in particular, considers ways to revitalize the knowledge base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs. *Critical Code* discusses four sets of questions: To what extent is software capability significant for the DoD? Is it becoming more or less significant and strategic in systems development? Will the advances in software producibility needed by the DoD emerge unaided from industry at a pace sufficient to meet evolving defense requirements? What are the opportunities for the DoD to make more effective use of emerging technology to improve software capability and software producibility? In which technology areas should the DoD invest in research to advance defense software capability and producibility?

To achieve consistent software project success under the pressures of today's software development environment, software organizations require achievable plans including viable estimates of schedule, resources, and risks. To estimate realistically, you must understand how to apply sound estimation processes, tools, and data. *Software Sizing*

In recent years, the systems designed to support activity in the fields of banking, health, transportation, space, aeronautics, defense, etc. have become increasingly larger and more complex. With the growing maturity of information and communication technologies, systems have been interconnected within growing networks, yielding new capabilities and services through the combination of system functionalities. This has led to a further increasing complexity that has to be managed in order to take advantage of these system integrations. The book is divided into two parts. The first part addresses the concept and practical illustrations of a "system of systems" and is a multidisciplinary introduction to the notion of a "systems of systems" that is discussed extensively in the current scientific and technical literature. After a critical comparison of the different definitions and a range of various practical illustrations, this part provides answers to key questions such as what a system of systems is and how its complexity can be mastered. The second part, described as "systems-of-systems engineering: methods and tools", focuses on both engineering and modeling, and standardization issues that are critical to deal with the key steps in systems of systems engineering: namely eliciting stakeholder needs, architecture optimization, integration of constituent systems, qualification, and utilization.

This book contains the refereed proceedings of the 14th International Conference on Agile Software Development, XP 2013, held in Vienna, Austria, in June 2013. In the last decade, the interest in agile and lean software development has been continuously growing. Agile and lean have evolved from a way of working -- restricted in the beginning to a few early adopters -- to the mainstream way of developing software. All this time, the XP conference series has actively promoted agility and widely disseminated research results in this area. XP 2013 successfully continued this tradition. The 17 full papers accepted for XP 2013 were selected from 52 submissions and are organized in sections on: teaching and learning; development teams; agile practices; experiences and lessons learned; large-scale projects; and architecture and design.

It is, indeed, widely acceptable today that nowhere is it more important to focus on the improvement of software quality than in the case of systems with requirements in the areas of safety and reliability - especially for distributed, real-time and embedded systems. Thus, much research work is under progress in these fields, since software process improvement impinges directly on achieved levels of quality, and many application experiments aim to show quantitative results demonstrating the efficacy of particular approaches. Requirements for safety and reliability - like other so-called non-functional requirements for computer-based systems - are often stated in imprecise and ambiguous terms, or not at all. Specifications focus on functional and technical aspects, with issues like safety covered only implicitly, or not addressed directly because they are felt to be obvious; unfortunately what is obvious to an end user or system user is progressively less so to others, to the extent that a software developer may not even be aware that safety is an issue. Therefore, there is a growing evidence for encouraging greater understanding of safety and reliability requirements issues, right across the spectrum from end user to software developer; not just in traditional safety-critical areas (e.g. nuclear, aerospace) but also acknowledging the need for such things as heart pacemakers and other medical and robotic systems to be highly dependable.

This book enhances learning about complex project management principles and practices through the introduction and discussion of a portfolio of tools presented as an evolving toolbox. Throughout the book, industry practitioners examine the toolsets that are part of the toolbox to develop a broader understanding of complex project management challenges and the available tools to address them. This approach establishes a dynamic, structured platform for a comprehensive analysis and assessment of the modern, rapidly changing, multifaceted business environment to teach the next generation of project managers to successfully cope with the ever increasing complexity of the 21st century.

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

The evolution of soft computing applications have offered a multitude of methodologies and techniques that are useful in facilitating new ways to address practical and real scenarios in a variety of fields. Exploring Innovative and Successful Applications of Soft Computing highlights the applications and conclusions associated with soft computing in different technological environments. Providing potential results based on new trends in the development of these services, this book aims to be a reference source for researchers, practitioners, and students interested in the most successful soft computing methods applied to recent problems.

The authoritative guide to the effective design and production of reliable technology products, revised and updated While most manufacturers have mastered the process of producing quality products, product reliability, software quality and software security has lagged behind. The revised second edition of Improving Product Reliability and Software Quality offers a comprehensive and detailed guide to implementing a hardware reliability and software quality process for technology products. The authors – noted experts in the field – provide useful tools, forms and spreadsheets for executing an effective product reliability and software quality development process and explore proven software quality and product reliability concepts. The authors discuss why so many companies fail after attempting to implement or improve their product reliability and software quality program. They outline the critical steps for implementing a successful program. Success hinges on establishing a reliability lab, hiring the right people and implementing a reliability and software quality process that does the right things well and works well together. Designed to be accessible, the book contains a decision matrix for small, medium and large companies. Throughout the book, the authors describe the hardware reliability and software quality process as well as the tools and techniques needed for putting it in place. The concepts, ideas and material presented are appropriate for any organization. This updated second edition: Contains new chapters on Software tools, Software quality process and software security. Expands the FMEA section to include software fault trees and software FMEAs. Includes two new reliability tools to accelerate design maturity and reduce the risk of premature wearout. Contains new material on preventative maintenance, predictive maintenance and Prognostics and Health Management (PHM) to better manage repair cost and unscheduled downtime. Presents updated information on reliability modeling and hiring reliability and software engineers. Includes a comprehensive review of the reliability process from a multi-disciplinary viewpoint including new material on uprating and counterfeit components.

Discusses aspects of competition, key quality and reliability concepts and presents the tools for implementation. Written for engineers, managers and consultants lacking a background in product reliability and software quality theory and statistics, the updated second edition of Improving Product Reliability and Software Quality explores all phases of the product life cycle.

Software is one of the most important products in human history and is widely used by all industries and all countries. It is also one of the most expensive and labor-intensive products in human history. Software also has very poor quality that has caused many major disasters and wasted many millions of dollars. Software is also the target of frequent and increasingly serious cyber-attacks. Among the reasons for these software problems is a chronic lack of reliable quantified data. This reference provides quantified data from many countries and many industries based on about 26,000 projects developed using a variety of methodologies and team experience levels. The data has been gathered between 1970 and 2017, so interesting historical trends are available. Since current average software productivity

and quality results are suboptimal, this book focuses on "best in class" results and shows not only quantified quality and productivity data from best-in-class organizations, but also the technology stacks used to achieve best-in-class results. The overall goal of this book is to encourage the adoption of best-in-class software metrics and best-in-class technology stacks. It does so by providing current data on average software schedules, effort, costs, and quality for several industries and countries. Because productivity and quality vary by technology and size, the book presents quantitative results for applications between 100 function points and 100,000 function points. It shows quality results using defect potential and DRE metrics because the number one cost driver for software is finding and fixing bugs. The book presents data on cost of quality for software projects and discusses technical debt, but that metric is not standardized. Finally, the book includes some data on three years of software maintenance and enhancements as well as some data on total cost of ownership.

Data Science for Software Engineering: Sharing Data and Models presents guidance and procedures for reusing data and models between projects to produce results that are useful and relevant. Starting with a background section of practical lessons and warnings for beginner data scientists for software engineering, this edited volume proceeds to identify critical questions of contemporary software engineering related to data and models. Learn how to adapt data from other organizations to local problems, mine privatized data, prune spurious information, simplify complex results, how to update models for new platforms, and more. Chapters share largely applicable experimental results discussed with the blend of practitioner focused domain expertise, with commentary that highlights the methods that are most useful, and applicable to the widest range of projects. Each chapter is written by a prominent expert and offers a state-of-the-art solution to an identified problem facing data scientists in software engineering. Throughout, the editors share best practices collected from their experience training software engineering students and practitioners to master data science, and highlight the methods that are most useful, and applicable to the widest range of projects. Shares the specific experience of leading researchers and techniques developed to handle data problems in the realm of software engineering Explains how to start a project of data science for software engineering as well as how to identify and avoid likely pitfalls Provides a wide range of useful qualitative and quantitative principles ranging from very simple to cutting edge research Addresses current challenges with software engineering data such as lack of local data, access issues due to data privacy, increasing data quality via cleaning of spurious chunks in data

Estimating Software-Intensive Systems Projects, Products, and Processes Pearson Education

This book constitutes the refereed proceedings of the First International Conference on Software Process, held in Minneapolis, MN, USA, in May 2007. The 28 revised full papers presented together with the abstracts of two keynote addresses cover process content, process tools and metrics, process management, process representation, analysis and modeling, experience report, and simulation modeling. Software development has been a troubling since it first started. There are seven chronic problems that have plagued it from the beginning: Incomplete and ambiguous user requirements that grow by >2% per month. Major cost and schedule overruns for large applications > 35% higher than planned. Low defect removal efficiency (DRE) Cancelled projects that are not completed: > 30% above 10,000 function points. Poor quality and low reliability after the software is delivered: > 5 bugs per FP. Breach of contract litigation against software outsource vendors. Expensive maintenance and enhancement costs after delivery. These are endemic problems for software executives, software engineers and software customers but they are not insurmountable. In Software Development Patterns and Antipatterns, software engineering and metrics pioneer Capers Jones presents technical solutions for all seven. The solutions involve moving from harmful patterns of software development to effective patterns of software development. The first section of the book examines common software development problems that have been observed in many companies and government agencies. The data on the problems comes from consulting studies, breach of contract lawsuits, and the literature on major software failures. This section considers the factors involved with cost overruns, schedule delays, canceled projects, poor quality, and expensive maintenance after deployment. The second section shows patterns that lead to software success. The data comes from actual companies. The section's first chapter on Corporate Software Risk Reduction in a Fortune 500 company was based on a major telecom company whose CEO was troubled by repeated software failures. The other chapters in this section deal with methods of achieving excellence, as well as measures that can prove excellence to C-level executives, and with continuing excellence through the maintenance cycle as well as for software development.

Instrument Engineers' Handbook, Third Edition: Volume Three: Process Software and Digital Networks provides an in-depth, state-of-the-art review of existing and evolving digital communications and control systems. While the book highlights the transportation of digital information by buses and networks, the total coverage doesn't stop there. It des

This report addresses Test and Evaluation (T&E) of software intensive systems and the DoD's efforts to improve the software process. DoD software costs total over \$30 billion a year, of which 2/3's is for maintaining, upgrading, and modifying operational systems already in production. Today's major defense systems depend largely on the quality of this complex and increasingly costly software. Because software error can cause a system to fail, possibly with life threatening consequences, software intensive systems need to be thoroughly tested before production. Charts and tables.

Designed to conform to the ISO/IEC standard 14143, the Common Software Measurement International Consortium (COSMIC) Function Point method has become the major estimation technique based on international standards for building software-intensive systems. COSMIC Function Points: Theory and Advanced Practices supplies a cutting-edge look at current and emerging practices in the international software measurement community. The editors have assembled an international panel of experts who detail the steps for measuring the functional size of software and developing project estimates with improved accuracy. They explain how to evaluate and compare systems to improve software reuse and development. Touching on the essential aspects of the next generation of functional size measurement methods, the book delineates best estimation and measurement practices as well as the development of benchmarks for quality improvement, including Six Sigma. This complete resource covers software measurement and estimation methods and practices for embedded systems, business applications, communications software, and control systems. Each chapter supplies the practical understanding required to create, implement, standardize, distribute, and adapt functional size measurement and project estimation to virtually any software context. Praise for: ... an excellent overview ... provides a strong knowledge background for both practitioners and researchers. ... With its broad background, it is useful for practically implementing and successfully adapting other functional sizing methods The COSMIC function point techniques presented in this book will help you to implement, master, and improve your estimation process. —Christof Ebert, Managing Director, Vector Consulting Services

The book is organized around basic principles of software project management: planning and estimating, measuring and controlling, leading and communicating, and managing

risk. Introduces software development methods, from traditional (hacking, requirements to code, and waterfall) to iterative (incremental build, evolutionary, agile, and spiral). Illustrates and emphasizes tailoring the development process to each project, with a foundation in the fundamentals that are true for all development methods. Topics such as the WBS, estimation, schedule networks, organizing the project team, and performance reporting are integrated, rather than being relegated to appendices. Each chapter in the book includes an appendix that covers the relevant topics from CMMI-DEV-v1.2, IEEE/ISO Standards 12207, IEEE Standard 1058, and the PMI® Body of Knowledge. (PMI is a registered mark of Project Management Institute, Inc.)

This book presents scientific interactions between the three interwoven and challenging areas of research and development of future ICT-enabled applications: software, complex systems and intelligent systems. Software intensive systems heavily interact with other systems, sensors, actuators, and devices, as well as other software systems and users. More and more domains involve software intensive systems, e.g. automotive, telecommunication systems, embedded systems in general, industrial automation systems and business applications. Moreover, web services offer a new platform for enabling software intensive systems. Complex systems research focuses on understanding overall systems rather than their components. Such systems are characterized by the changing environments in which they act, and they evolve and adapt through internal and external dynamic interactions. The development of intelligent systems and agents features the use of ontologies, and their logical foundations provide a fruitful impulse for both software intensive systems and complex systems. Research in the field of intelligent systems, robotics, neuroscience, artificial intelligence, and cognitive sciences is a vital factor in the future development and innovation of software intensive and complex systems.

This unique volume explores cutting-edge management approaches to developing complex software that is efficient, scalable, sustainable, and suitable for distributed environments. Practical insights are offered by an international selection of pre-eminent authorities, including case studies, best practices, and balanced corporate analyses. Emphasis is placed on the use of the latest software technologies and frameworks for life-cycle methods, including the design, implementation and testing stages of software development. Topics and features: · Reviews approaches for reusability, cost and time estimation, and for functional size measurement of distributed software applications · Discusses the core characteristics of a large-scale defense system, and the design of software project management (SPM) as a service · Introduces the 3PR framework, research on crowdsourcing software development, and an innovative approach to modeling large-scale multi-agent software systems · Examines a system architecture for ambient assisted living, and an approach to cloud migration and management assessment · Describes a software error proneness mechanism, a novel Scrum process for use in the defense domain, and an ontology annotation for SPM in distributed environments · Investigates the benefits of agile project management for higher education institutions, and SPM that combines software and data engineering This important text/reference is essential reading for project managers and software engineers involved in developing software for distributed computing environments. Students and researchers interested in SPM technologies and frameworks will also find the work to be an invaluable resource. Prof. Zaigham Mahmood is a Senior Technology Consultant at Debasis Education UK and an Associate Lecturer (Research) at the University of Derby, UK. He also holds positions as Foreign Professor at NUST and IIU in Islamabad, Pakistan, and Professor Extraordinaire at the North West University Potchefstroom, South Africa.

The Software Error Estimate Program (SWEEP) aids you in the management and prediction of errors and defects in software intensive systems. It will support you in establishing goals for error detection during software development and can help to track progress against those goals. You can also use SWEEP to monitor and help control the quality of software products by predicting the number of defects remaining in your software system, or latent error content. You use the latent error content to estimate the amount of testing and repair effort needed to reach different quality levels and to understand the quality of a delivered product. SWEEP can also help you project software outages due to software related defects in software intensive systems. SWEEP supports a variety of different organizational approaches to monitoring defects during software development. Organizations that measure and track defects during the software implementation phase can iterate through development and test intervals, using SWEEP to predict the latent error content. Organizations that measure and track in one phase based upon the data collected in previous phases. SWEEP can also assist you in the planning of software projects. You enter the number of defects (per code unit) and the estimated latent error content to derive a profile pattern for error discovery that will project the number of remaining errors on a phase-by-phase basis. (AN).

[Copyright: e837b593584f076571e9f42244b48559](https://doi.org/10.1007/978-1-4939-9835-9)