

Managing The Software Process Watts S Humphrey

Well-known author and long-time manager Watts Humphrey offers keen insight into the special challenge of identifying, motivating, and organizing creative technical people, and the opportunities involved in managing these people.

The SEI's Capability Maturity Model (CMM) has been widely adopted by companies seeking enhanced quality and heightened productivity in software development. This guide provides detailed instruction on how to put this model into practice and thereby raise an organization to the next level. Templates, sample documents and presentation materials are included on the CD-ROM.

Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, Mary Beth Chrissis "In every sense, the CMM represents the best thinking in the field today... this book is targeted at anyone involved in improving the software process, including members of assessment or evaluation teams, members of software engineering process groups, software managers, and software practitioners..." From the Foreword by Watts Humphrey The Capability Maturity Model for Software (CMM) is a framework that demonstrates the key elements of an effective software process. The CMM describes an evolutionary improvement path for software development from an ad hoc, immature process to a mature, disciplined process, in a path laid out in five levels. When using the CMM, software professionals in government and industry can develop and improve their ability to identify, adopt, and use sound management and technical practices for delivering quality software on schedule and at a reasonable cost. This book provides a description and technical overview of the CMM, along with guidelines for improving software process management overall. It is a sequel to Watts Humphrey's important work, *Managing the Software Process*, in that it structures the maturity framework presented in that book more formally.

Features: Compares the CMM with ISO 9001 Provides an overview of ISO's SPICE project, which is developing international standards for software process improvement and capability determination Presents a case study of IBM Houston's Space Shuttle project, which is frequently referred to as being at Level 5 0201546647B04062001

This book aims to give you a head start by providing a detailed down-to-earth account of how one Swedish company implemented Scrum and XP with a team of approximately 40 people and how they continuously improved their process over a year's time. Under the leadership of Henrik Kniberg they experimented with different team sizes, different sprint lengths, different ways of defining "done," different formats for product backlogs and sprint backlogs, different testing strategies, different ways of doing demos, different ways of synchronizing multiple Scrum teams, etc. They also experimented with XP practices - different ways of doing continuous build, pair programming, test driven development, etc, and how to combine this with Scrum. This second edition is an annotated version, a "director's cut" where Henrik reflects upon the content and shares new insights gained since the first version of the book.

A benchmark text on software development and quantitative softwareengineering "We all trust software. All too frequently, this trust is misplaced. Larry Bernstein has created and applied quantitative techniques to develop trustworthy software systems. He and C. M. Yuhas have organized this quantitative experience into a book of great value to make software trustworthy for all of us." -Barry Boehm Trustworthy Systems Through Quantitative Software Engineering proposes a novel, reliability-driven software engineering approach, and discusses human factors in software engineering and how these affect team dynamics. This practical approach gives softwareengineering students and professionals a solid foundation in problem analysis, allowing them to meet customers' changing needs by tailoring their projects to meet specific challenges, and complete projects on schedule and within budget. Specifically, it helps developers identify customer requirements, develop software designs, manage a software development team, and evaluate software products to customer specifications. Students learn "magic numbers of software engineering," rules of thumb that show how to simplify architecture, design, and implementation. Case histories and exercises clearly present successful softwareengineers' experiences and illustrate potential problems, results, and trade-offs. Also featuring an accompanying Web site with additional and related material, Trustworthy Systems Through Quantitative Software Engineering is a hands-on, project-oriented resource for upper-level software and computer science students, engineers, professional developers, managers, and professionals involved in software engineering projects. An Instructor's Manual presenting detailed solutions to all the problems in the book is available from the Wiley editorial department. An Instructor Support FTP site is also available.

This newest book from Watts Humphrey is a hands-on introduction to basic disciplines of software engineering. Designed as a workbook companion to any introductory programming or software-engineering text, Humphrey provides here the practical means to integrate his highly regarded Personal Software Process (PSP) into the undergraduate curriculum. Applying the book's exercises to course assignments, students learn both to manage their time effectively and to monitor the quality of their work, good practices they will need to be successful in their future careers. The book is supported by its own electronic supplement, which includes spreadsheets for data entry and analysis. A complete instructor's package is also available. By mastering PSP techniques early in their studies, students can avoid-or overcome-the popular "hacker" ethic that leads to so many bad habits. Employers will appreciate new hires prepared to do competent professional work without, as now is common, expensive retraining and years of experience.

An industry insider explains why there is so much bad software—and why academia doesn't teach programmers what industry wants them to know. Why is software so prone to bugs? So vulnerable to viruses? Why are software products so often delayed, or even canceled? Is software development really hard, or are software developers just not that good at it? In *The Problem with Software*, Adam Barr examines the proliferation of bad software, explains what causes it, and offers some suggestions on how to improve the situation. For one thing, Barr points out, academia doesn't teach programmers what they actually need to know to do their jobs: how to work in a team to create code that works reliably and can be maintained by somebody other than the original authors. As the size and complexity of commercial software have grown, the gap between academic computer science and industry has widened. It's an open secret that there is little engineering in software engineering, which continues to rely not on codified scientific knowledge but on intuition and experience. Barr, who worked as a programmer for more than twenty years, describes how the industry has evolved, from the era of mainframes and Fortran to today's embrace of the cloud. He explains bugs and why software has so many of them, and why today's interconnected computers offer fertile ground for viruses and worms. The difference between good and bad software can be a single line of code, and Barr includes code to

illustrate the consequences of seemingly inconsequential choices by programmers. Looking to the future, Barr writes that the best prospect for improving software engineering is the move to the cloud. When software is a service and not a product, companies will have more incentive to make it good rather than “good enough to ship.”

The one resource needed to create reliable software This text offers a comprehensive and integrated approach to software quality engineering. By following the author's clear guidance, readers learn how to master the techniques to produce high-quality, reliable software, regardless of the software system's level of complexity. The first part of the publication introduces major topics in software quality engineering and presents quality planning as an integral part of the process. Providing readers with a solid foundation in key concepts and practices, the book moves on to offer in-depth coverage of software testing as a primary means to ensure software quality; alternatives for quality assurance, including defect prevention, process improvement, inspection, formal verification, fault tolerance, safety assurance, and damage control; and measurement and analysis to close the feedback loop for quality assessment and quantifiable improvement. The text's approach and style evolved from the author's hands-on experience in the classroom. All the pedagogical tools needed to facilitate quick learning are provided: * Figures and tables that clarify concepts and provide quick topic summaries * Examples that illustrate how theory is applied in real-world situations * Comprehensive bibliography that leads to in-depth discussion of specialized topics * Problem sets at the end of each chapter that test readers' knowledge This is a superior textbook for software engineering, computer science, information systems, and electrical engineering students, and a dependable reference for software and computer professionals and engineers.

Most modern software development projects require teams, and good teamwork largely determines a project's success. The Team Software Process (TSP), created by Watts S. Humphrey, is a set of engineering practices and team concepts that produce effective teams, thereby helping developers deliver high-quality products on time and within budget. TSP bridges Humphrey's seminal work on the Capability Maturity Model (CMM), an improvement framework for the entire software organization, and his Personal Software Process (PSP), practices designed to improve the work of individual developers. Typical first-time TSP teams increase productivity by more than 50 percent while greatly increasing the quality of their delivered products. However, TSP teams only continue to improve under the guidance of a capable coach. One industrial-strength team, for example, increased its productivity by an additional 94 percent and reduced test defects by 85 percent through three consecutive TSP quarterly product release cycles. Without competent coaching, teams often do not progress much beyond the initial one-time improvement seen after the introduction of the TSP. Humphrey distinguishes between TSP coaching and TSP leadership, explaining why the skillful performance of both functions is critical. In this practical guide, he shares coaching methods that have repeatedly inspired TSP teams and steered them toward success. With the help of a coach, TSP teams undergo a brief but intense project launch in which they define their own processes, make their own plans, and negotiate their commitments with management, resulting in dramatically enhanced performance. Whether you are considering the TSP or are actively implementing it, TSPSM—Coaching Development Teams provides the invaluable examples, guidelines, and suggestions you need to get started and keep developing as a team coach. It's meant to complement Humphrey's other books, TSPSM—Leading a Development Team and PSPSM: A Self-Improvement Process for Software Engineers. Together, the three works offer a rich resource for improving your software development capabilities.

The author, drawing on years of experience at IBM and the SEI, provides here practical guidance for improving the software development and maintenance process. He focuses on understanding and managing the software process because this is where he feels organizations now encounter the most serious problems, and where he feels there is the best opportunity for significant improvement. Both program managers and practicing programmers, whether working on small programs or large-scale projects, will learn how good their own software process is, how they can make their process better, and where they need to begin. "This book will help you move beyond the turning point, or crisis, of feeling over-whelmed by the task of managing the software process to understanding what is essential in software management and what you can do about it." Peter Freeman, from the Foreword 0201180952B04062001

Publisher Fact Sheet A concise, hands-on approach to managing & improving the critical requirements process in software development.

A Lifetime of Invaluable Management Insights from Legendary Software Quality Guru Watts S. Humphrey In 1986, Watts S. Humphrey made an outrageous commitment: a promise to transform software development. As the pioneering innovator behind SEI's Capability Maturity Model (CMM), Personal Software Process (PSP), and Team Software Process (TSP), Humphrey has more than met that promise. But his contributions go beyond methodology: For decades, his deeply personal writings on project management have been admired by software engineers worldwide. Reflections on Management brings together Humphrey's best and most influential essays and articles--sharing insights that will be indispensable for anyone who must achieve superior results in software or any other endeavor. Collected here for the first time, these works offer compelling insights into everything from planning day-to-day work to improving quality, encouraging teamwork to becoming a truly great leader. All of these writings share a powerful vision, grounded by a life in software that has extended across nearly six decades. The vision is this: To succeed, professionals must effectively manage for more than plans, schedules, and code--they must manage teams, bosses, and above all, themselves.

“Collaboration Explained is a deeply pragmatic book that helps agile practitioners understand and manage complex organizational and team dynamics. As an agile coach, I've found the combination of straightforward advice and colorful anecdotes to be invaluable in guiding and focusing interactions with my teams. Jean's wealth of experience is conveyed in a carefully struck balance of reference guides and prose, facilitating just-in-time learning in the agile spirit. All in all, a superb resource for building stronger teams that's fit for agile veterans and neophytes alike.” —Arlen Bankston, Lean Agile Practice Manager, CC Pace “If Agile is the new ‘what,’ then surely Collaboration is the new ‘how.’ There are many things I really like about Jean's new book. Right at the top of the list is that I don't have to make lists of ideas for collaboration and facilitation anymore. Jean has it all. Not only does she have those great ideas for meetings, retrospectives, and team decision-making that I need to remember, but the startling new and thought-provoking ideas are there too. And the stories, the stories, the stories! The best way to transfer wisdom. Thanks, Jean!” —Linda Rising, Independent Consultant The Hands-On Guide to Effective Collaboration in Agile Projects To succeed, an agile project demands outstanding collaboration among all its stakeholders. But great collaboration doesn't happen by itself; it must be carefully planned and facilitated throughout the entire project lifecycle. Collaboration Explained is the first book to bring together proven, start-to-finish techniques for ensuring effective collaboration in any agile software project. Since the early days of the agile movement, Jean Tabaka has been studying and promoting collaboration in agile environments. Drawing on her unsurpassed experience, she offers clear guidelines and easy-to-use collaboration templates for every significant project event: from iteration and release planning, through project chartering, all the way through post-project retrospectives. Tabaka's hands-on techniques are applicable to every leading agile methodology, from Extreme Programming and Scrum to Crystal Clear. Above all, they are practical: grounded in a powerful understanding of the technical, business, and human challenges you face as a project manager or development team member. · Build collaborative software development cultures, leaders, and teams · Prepare yourself to collaborate—and prepare your team · Define clear roles for each participant in promoting collaboration · Set your collaborative agenda · Master tools for organizing collaboration more efficiently · Run effective collaborative meetings—including brainstorming sessions · Promote better small-group and pair-programming collaboration · Get better information, and use it to make better decisions · Use non-abusive

conflict to drive positive outcomes · Collaborate to estimate projects and schedules more accurately · Strengthen collaboration across distributed, virtual teams · Extend collaboration from individual projects to the entire development organization

Most software-development groups have embarrassing records: By some accounts, more than half of all software projects are significantly late and over budget, and nearly a quarter of them are cancelled without ever being completed. Although developers recognize that unrealistic schedules, inadequate resources, and unstable requirements are often to blame for such failures, few know how to solve these problems. Fortunately, the Personal Software Process (PSP) provides a clear and proven solution. Comprising precise methods developed over many years by Watts S. Humphrey and the Software Engineering Institute (SEI), the PSP has successfully transformed work practices in a wide range of organizations and has already produced some striking results. This book describes the PSP and is the definitive guide and reference for its latest iteration. PSP training focuses on the skills required by individual software engineers to improve their personal performance. Once learned and effectively applied, PSP-trained engineers are qualified to participate on a team using the Team Software Process (TSP), the methods for which are described in the final chapter of the book. The goal for both PSP and TSP is to give developers exactly what they need to deliver quality products on predictable schedules. PSPSM: A Self-Improvement Process for Software Engineers presents a disciplined process for software engineers and anyone else involved in software development. This process includes defect management, comprehensive planning, and precise project tracking and reporting. The book first scales down industrial software practices to fit the needs of the module-sized program development, then walks readers through a progressive sequence of practices that provide a sound foundation for large-scale software development. By doing the exercises in the book, and using the PSP methods described here to plan, evaluate, manage, and control the quality of your own work, you will be well prepared to apply those methods on ever larger and more critical projects. Drawing on the author's extensive experience helping organizations to achieve their development goals, and with the PSP benefits well illustrated, the book presents the process in carefully crafted steps. The first chapter describes overall principles and strategies. The next two explain how to follow a defined process, as well as how to gather and use the data required to manage a programming job. Several chapters then cover estimating and planning, followed by quality management and design. The last two chapters show how to put the PSP to work, and how to use it on a team project. A variety of support materials for the book, as described in the Preface, are available on the Web. If you or your organization are looking for a way to improve your project success rate, the PSP could well be your answer.

Watts Humphrey, inventor of CMM, PSP, and TSP provides team leaders with a whole new way of leading an effective development team.

"Every senior executive needs to read this book." --Robert Musson Vice President, Business Strategy Cenus Technologies "An informative book for any business person (not just technologists) who has ever been associated or involved with a software development effort and thought 'there must be a better way!' Watts has provided that better way-- the PSP/TSP, and a great book." --Roy Kinkaid, Head of Continuous Improvement and Software Quality Assurance, EBS Dealing Resources Watts Humphrey is the well-known author of methods and models widely used by organizations, teams, and individuals to improve the efficiency and effectiveness of software development. In *Winning with Software*, he shows corporate executives and senior managers why software is both a business problem and a business opportunity. "This book is extremely well written and targets the right audience. I plan to buy a copy for each of my executives." --Kevin J. Berk, Director, Process Improvement, Total Quality Systems Humphrey, drawing on his own extensive executive and management experience, first demonstrates the critical importance of software to nearly every business, large and small. He then outlines seven steps needed to gain control of a software operation and transform it into a professional, businesslike engineering function. Failure to recognize the importance of software, and to take charge of its development process, runs the risk of damaging the entire business. By contrast, Humphrey relates the substantial benefits real organizations have obtained from such awareness and control, and he concludes with an analysis of the impressive financial returns the recommended transformations typically yield. "This is a great book that will play a valuable role. It has excellent anecdotes that illustrate the points being made, as well as good examples depicting the problems faced by teams and managers. I look forward to sharing it with my colleagues." --Steven Sliwa, President & CEO, Insitu Group Inc. and former President of Embry-Riddle University "The logical approach, the high level explanations, and the application of real-life experiences make the book not only credible but easily understood. If a large number of CEOs don't at least try out the book's concepts, I will be greatly surprised." --David Webb Software Engineering Project Manager, Hill Air Force Base

The IEEE's Software Engineering standards are the internationally accepted guidelines for developing software in the commercial, government, and private sectors. This text presents a Software Life-Cycle Model to complement the standards and aid development.

A classic treatise that defined the field of applied demand analysis, *Consumer Demand in the United States: Prices, Income, and Consumption Behavior* is now fully updated and expanded for a new generation. Consumption expenditures by households in the United States account for about 70% of America's GDP. The primary focus in this book is on how households adjust these expenditures in response to changes in price and income. Econometric estimates of price and income elasticities are obtained for an exhaustive array of goods and services using data from surveys conducted by the Bureau of Labor Statistics, providing a better understanding of consumer demand. Practical models for forecasting future price and income elasticities are also demonstrated. Fully revised with over a dozen new chapters and appendices, the book revisits the original Taylor-Houthakker models while examining new material as well, such as the use of quantile regression and the stationarity of consumer preference. It also explores the emerging connection between neuroscience and consumer behavior, integrating the economic literature on demand theory with psychology literature. The most comprehensive treatment of the topic to date, this volume will be an essential resource for any researcher, student or professional economist working on consumer behavior or demand theory, as well as investors and policymakers concerned with the impact of economic fluctuations.

CMMI® for Services (CMMI-SVC) is a comprehensive set of guidelines to help organizations establish and improve processes for delivering services. By adapting and extending proven standards and best practices to reflect the unique challenges faced in service industries, CMMI-SVC offers providers a practical and focused framework for achieving higher levels of service quality, controlling costs, improving schedules, and ensuring user satisfaction. A member of the newest CMMI model, CMMI-SVC Version 1.3, reflects changes to the model made for all constellations, including clarifications of high-maturity practices, alignment of the sixteen core process areas, and improvements in the SCAMPI appraisal method. The indispensable CMMI® for Services, Second Edition, is both an introduction to the CMMI-SVC model and an authoritative reference for it. The contents include the complete model itself, formatted for quick reference. In addition, the book's authors have refined the model's introductory chapters; provided marginal notes to clarify the nature of particular process areas and to show why their practices are valuable; and inserted longer sidebars to explain important concepts. Brief essays by people with experience in different application areas further illustrate how the model works in practice and what benefits it offers. The book is divided into three parts. Part One begins by thoroughly explaining CMMI-SVC, its concepts, and its use. The authors provide robust information about

service concepts, including a discussion of lifecycles in service environments; outline how to start using CMMI-SVC; explore how to achieve process improvements that last; and offer insights into the relationships among process areas. Part Two describes generic goals and practices, and then details the complete set of twenty-four CMMI-SVC process areas, including specific goals, specific practices, and examples. The process areas are organized alphabetically by acronym and are tabbed for easy reference. Part Three contains several useful resources, including CMMI-SVC-related references, acronym definitions, a glossary of terms, and an index. Whether you are new to CMMI models or are already familiar with one or more of them, this book is an essential resource for service providers interested in learning about or implementing process improvement.

Pioneering software engineer Capers Jones has written the first and only definitive history of the entire software engineering industry. Drawing on his extraordinary vantage point as a leading practitioner for several decades, Jones reviews the entire history of IT and software engineering, assesses its impact on society, and previews its future. One decade at a time, Jones assesses emerging trends and companies, winners and losers, new technologies, methods, tools, languages, productivity/quality benchmarks, challenges, risks, professional societies, and more. He quantifies both beneficial and harmful software inventions; accurately estimates the size of both the US and global software industries; and takes on "unexplained mysteries" such as why and how programming languages gain and lose popularity.

Based on the needs of the educational community, and the software professional, this book takes a unique approach to teaching software testing. It introduces testing concepts that are managerial, technical, and process oriented, using the Testing Maturity Model (TMM) as a guiding framework. The TMM levels and goals support a structured presentation of fundamental and advanced test-related concepts to the reader. In this context, the interrelationships between theoretical, technical, and managerial concepts become more apparent. In addition, relationships between the testing process, maturity goals, and such key players as managers, testers and client groups are introduced. Topics and features: - Process/engineering-oriented text - Promotes the growth and value of software testing as a profession - Introduces both technical and managerial aspects of testing in a clear and precise style - Uses the TMM framework to introduce testing concepts in a systematic, evolutionary way to facilitate understanding - Describes the role of testing tools and measurements, and how to integrate them into the testing process Graduate students and industry professionals will benefit from the book, which is designed for a graduate course in software testing, software quality assurance, or software validation and verification Moreover, the number of universities with graduate courses that cover this material will grow, given the evolution in software development as an engineering discipline and the creation of degree programs in software engineering.

TSPi overview; The logic of the team software process; The TSPi process; The team roles; Using the TSPi; Teamwork.

bull; The must-have reference for every technical writer, editor, and documentation manager bull; Provides all the information you need to document hardware, software, or other computer products bull; Written by award-winning documentation experts at Sun Technical Publications, Read Me First! is the most comprehensive guide to creating documentation that is clear, consistent, and easy to understand

"While it is usually helpful to launch improvement programs, many such programs soon get bogged down in detail. They either address the wrong problems, or they keep beating on the same solutions, wondering why things don't improve. This is when you need an objective way to look at the problems. This is the time to get some data." Watts S. Humphrey, from the Foreword This book, drawing on work done at the Software Engineering Institute and other organizations, shows how to use measurements to manage and improve software processes. The authors explain specifically how quality characteristics of software products and processes can be quantified, plotted, and analyzed so the performance of software development activities can be predicted, controlled, and guided to achieve both business and technical goals. The measurement methods presented, based on the principles of statistical quality control, are illuminated by application examples taken from industry. Although many of the methods discussed are applicable to individual projects, the book's primary focus is on the steps software development organizations can take toward broad-reaching, long-term success. The book particularly addresses the needs of software managers and practitioners who have already set up some kind of basic measurement process and are ready to take the next step by collecting and analyzing software data as a basis for making process decisions and predicting process performance. Highlights of the book include: Insight into developing a clear framework for measuring process behavior Discussions of process performance, stability, compliance, capability, and improvement Explanations of what you want to measure (and why) and instructions on how to collect your data Step-by-step guidance on how to get started using statistical process control If you have responsibilities for product quality or process performance and you are ready to use measurements to manage, control, and predict your software processes, this book will be an invaluable resource.

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.

This newest book from Watts Humphrey is a hands-on introduction to basic disciplines of software engineering. Designed as a workbook companion to any introductory programming or software-engineering text, Humphrey provides here the practical means to integrate his highly regarded Personal Software Process (PSP) into college and university curricula. The book may also be adapted for use in industrial training or for self-improvement by practicing software engineers. Applying the book's exercises to their course assignments, students learn both to manage their time effectively and to monitor the quality of their work, good practices they will need to be successful in their future careers. The book is supported by its own electronic supplement, which includes spreadsheets for data entry and analysis. A complete instructor's package is also available. By mastering PSP techniques early in their studies, students can avoid--or overcome--the popular "hacker" ethic that leads to so many bad habits. Employers will appreciate new hires prepared to do competent professional work without, as now is common, expensive retraining and years of experience.

Managing the Software Process Addison-Wesley Professional

CERT® Resilience Management Model (CERT-RMM) is an innovative and transformative way to manage operational resilience in complex, risk-evolving environments. CERT-RMM distills years of research into best practices for managing the security and survivability of people, information, technology, and facilities. It integrates these best practices into a unified, capability-focused maturity model that encompasses security, business continuity, and IT operations. By using CERT-RMM, organizations can escape silo-driven approaches to managing operational risk and align to achieve strategic resilience management goals. This book both introduces CERT-RMM and presents the model in its entirety. It begins with essential background for all professionals, whether they have previously used process

improvement models or not. Next, it explains CERT-RMM's Generic Goals and Practices and discusses various approaches for using the model. Short essays by a number of contributors illustrate how CERT-RMM can be applied for different purposes or can be used to improve an existing program. Finally, the book provides a complete baseline understanding of all 26 process areas included in CERT-RMM. Part One summarizes the value of a process improvement approach to managing resilience, explains CERT-RMM's conventions and core principles, describes the model architecturally, and shows how it supports relationships tightly linked to your objectives. Part Two focuses on using CERT-RMM to establish a foundation for sustaining operational resilience management processes in complex environments where risks rapidly emerge and change. Part Three details all 26 CERT-RMM process areas, from asset definition through vulnerability resolution. For each, complete descriptions of goals and practices are presented, with realistic examples. Part Four contains appendices, including Targeted Improvement Roadmaps, a glossary, and other reference materials. This book will be valuable to anyone seeking to improve the mission assurance of high-value services, including leaders of large enterprise or organizational units, security or business continuity specialists, managers of large IT operations, and those using methodologies such as ISO 27000, COBIT, ITIL, or CMMI.

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.

"Every business is a software business, and every business can profit from improved software processes" "Leadership, Teamwork, and Trust" discusses the critical importance of knowledge work to the success of modern organizations. It explains concrete and necessary steps for reshaping the way in which software development, specifically, is conducted. A sequel to Humphrey's influential "Winning with Software," this book presents new and copious data to reinforce his widely adopted methods for transforming knowledge work into a significant and sustainable competitive advantage, thereby realizing remarkable returns. Humphrey addresses here the broader business community--executives and senior managers who must recognize that today, every business is a software business.

CMMI® for Development (CMMI-DEV) describes best practices for the development and maintenance of products and services across their lifecycle. By integrating essential bodies of knowledge, CMMI-DEV provides a single, comprehensive framework for organizations to assess their development and maintenance processes and improve performance. Already widely adopted throughout the world for disciplined, high-quality engineering, CMMI-DEV Version 1.3 now accommodates other modern approaches as well, including the use of Agile methods, Lean Six Sigma, and architecture-centric development. CMMI® for Development, Third Edition, is the definitive reference for CMMI-DEV Version 1.3. The authors have revised their tips, hints, and cross-references, which appear in the margins of the book, to help you better understand, apply, and find information about the content of each process area. The book includes new and updated perspectives on CMMI-DEV in which people influential in the model's creation, development, and transition share brief but valuable insights. It also features four new case studies and five contributed essays with practical advice for adopting and using CMMI-DEV. This book is an essential resource—whether you are new to CMMI-DEV or are familiar with an earlier version—if you need to know about, evaluate, or put the latest version of the model into practice. The book is divided into three parts. Part One offers the broad view of CMMI-DEV, beginning with basic concepts of process improvement. It introduces the process areas, their components, and their relationships to each other. It describes effective paths to the adoption and use of CMMI-DEV for process improvement and benchmarking, all illuminated with fresh case studies and helpful essays. Part Two, the bulk of the book, details the generic goals and practices and the twenty-two process areas now comprising CMMI-DEV. The process areas are organized alphabetically by acronym for easy reference. Each process area includes goals, best practices, and examples. Part Three contains several useful resources, including CMMI-DEV-related references, acronym definitions, a glossary of terms, and an index.

This new work from Watts Humphrey, author of the influential book, *Managing the Software Process*, broadens his orderly view of software process management, and lays the foundation for a disciplined approach to software engineering. In his earlier book, the author developed concrete methods for managing software development and maintenance. These methods, now commonly practiced in industry, provide programmers and managers with specific steps they can take to evaluate and improve their software capabilities. In this new book, Humphrey scales those methods down to a personal level, helping software engineers develop the skills and habits needed to plan, track, and analyze large, complex projects. Humphrey and others have used material from this book to train professionals and students around the world in a projects-oriented software engineering course. First establishing the need for discipline in software engineering, and the benefits to practitioners of learning how to manage their personal software process, Humphrey then develops a model that they can use to monitor, test, and improve their work. Examples drawn from industry enhance the practical focus of the book, while project exercises give readers the opportunity to practice software process management as they learn it. Features: presents concepts and methods for a disciplined software engineering process; scales down industrial practices for planning, tracking, analysis, and defect management to fit the needs of small-scale program development; and shows how small project disciplines provide a solid base for larger projects.

This informative book is designed to help professionals involved with development of software or systems manage process improvement initiatives within their company by explaining the history, method and psychology behind AFA.

Glass explores a critical, yet strangely neglected, question: What is the role of creativity in software engineering and computer programming? With his trademark easy-to-read style and practical approach, backed by research and personal experience, Glass takes on a wide range of related angles and implications. (Computer Books)

"Mantle and Lichty have assembled a guide that will help you hire, motivate, and mentor a software development team that functions at the highest level. Their rules of thumb and coaching advice are great blueprints for new and experienced software engineering managers alike." —Tom Conrad, CTO, Pandora "I wish I'd had this material available years ago. I see lots and lots of 'meat' in here that I'll use over and over again as I try to become a better manager. The writing style is right on, and I love the personal anecdotes." —Steve Johnson, VP, Custom Solutions, DigitalFish All too often, software development is deemed unmanageable. The news is filled with stories of projects that have run catastrophically over schedule and budget. Although adding some formal discipline to the development process has improved the situation, it has by no means solved the problem. How can it be, with so much time and money spent to get software development under control, that it remains so unmanageable? In *Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams*, Mickey W. Mantle and Ron Lichty answer that persistent question with a simple observation: You first must make programmers and software teams manageable. That is, you need to begin by understanding your people—how to hire them, motivate them, and lead them to develop and deliver great products. Drawing on their combined seventy years of software development and management experience, and highlighting the insights and wisdom of other successful managers, Mantle and Lichty provide the guidance you need to manage people and teams in order to deliver software successfully. Whether you are new to software management, or have already been working in that role, you will appreciate the real-world knowledge and practical tools packed into this guide.

Software Project Management explains the latest management strategies and techniques in software developments. It covers such issues as keeping the team motivated, cost-justifying strategies, deadlines and budgets.

Project initiation; Project planning; Project execution and termination.

[Copyright: 37d294571fbafb3a7e17b8fbedf2f702](https://www.pdfdrive.com/managing-the-software-process-watts-s-humphrey-pdf/ebook/37d294571fbafb3a7e17b8fbedf2f702)