Managing The Software Process

Shipping imperfect software is like going into debt. When you incur debt, the illusion of doing things faster can lead to exponential growth in the cost of maintaining software. Software debt takes five major forms: technical, quality, configuration management, design, and platform experience. In today's rush to market, software debt is inevitable. And that's okay—if you're careful about the debt you incur, and if you quickly pay it back. In Managing Software Debt, leading Agile expert Chris Sterling shows how understanding software debt can help you move products to market faster, with a realistic plan for refactoring them based on experience. Writing for all Agile software professionals, Sterling explains why you're going into software debt whether you know it or not—and why the interest on that debt can bring projects to a standstill. Next, he thoroughly explains each form of software debt, showing how to plan for it intelligently and repay it successfully. You'll learn why accepting software debt is not the same as deliberate sloppiness, and you'll learn how to use the software debt concept to systematically improve architectural agility. Coverage includes Managing tensions between speed and perfection and recognizing that you'll inevitably ship some "not quite right" code Planning to minimize interest payments by paying debts quickly Building architectures that respond to change and help enterprises run more smoothly Incorporating emergent architecture concepts into daily activities, using Agile collaboration and refactoring techniques Delivering code and other software internals that reduce the friction of future change Using early, automated testing to move past the "break/fix" mentality Scripting and streamlining both deployment and rollback Implementing team configuration patterns and knowledge sharing approaches that make software debt easier to repay Clearing away technical impediments in existing architectures Using the YAGNI ("you ain't gonna need it") approach to strip away unnecessary complexity Using this book's techniques, senior software leadership can deliver more business value; managers can organize and support development teams more effectively; and teams and team members can improve their performance throughout the development lifecycle.

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 relegating 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.)

Partial Contents1: Software Ascents- Components of Adaptive Software Development2: Thriving at the Edge of Chaos- The Adaptive Development Model3: The Project Mission- Identify the Mission- Create Mission Artifacts- Share Mission Values- Focus on Results4: Planning Adaptive Development Cycles- Adaptive Planning Techniques- The Evolving World of Components5: Great Groups and the Ability to Collaborate- Using Complexity Concepts to Improve Collaboration- Joint Application Development6: Learning: Models, Techniques, and Cycle Review Practices- Software Inspections- Project Postmortems7: Why Even Good Managers Cause Projects to Fail- Disruptive Technologies- No Silver Bullet8: Adaptive Management- The Progression from Process to Pattern9: Workstate Life Cycle Management- Managing Workflow in an Adaptive Environment10: Structural Collaboration- Eight Guidelines for Applying Rigor to Project Work11: Managing Project Time Cycles- Plan the Project12: Dawdling, McLuhan, and Thin Air- Organizational Growth- Surviving in Thin AirBibliographyIndex

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.

Biting and Humorous Tales of a Software Engineering Manager
Management of the Object-oriented Development Process
Managing Systems and IT Projects
Managing Technical People
Introduction to Software Project Management
Approaches and Tools for Practical Development
A Discipline for Software Engineering

"This book consists of a series of high-level discussions on technical and managerial issues related to object-oriented development"--Provided by publisher.

Although software development is one of the most complex activities carried out by man, sound development processes and proper project management can help ensure your software projects are delivered on time and under budget. Providing the know-how to manage software projects effectively, Introduction to Software Project Management supplies an accessible introduction to software project management. The book begins with an overview of the fundamental techniques of project management and the technical aspects of software development. This section supplies the understanding of the techniques required to mitigate uncertainty in projects and better control the complexity of software development projects. The second part illustrates the technical activities of software development in a coherent process—describing how to customize this process to fit a wide range of software development scenarios. Examines project management frameworks and software development standards, including ESA and NASA guidelines, PRINCE2®, and PMBOK® Addresses open source development practices and tools so readers can adopt best practices and get started with tools that are available for free Explains how to tailor the development process to different kinds of products and formalities, including the development of web applications Includes access to additional material for both practitioners and teachers at www.spmbook.com Supplying an analysis of existing development and management frameworks, the book describes how to set up an open-source tool infrastructure to manage projects. Since practitioners must be able to mix traditional and agile techniques effectively, the book covers both and explains how to use traditional techniques for planning and developing software components alongside agile methodologies. It does so in a manner that will help you to foster freedom and creativity in assembling the processes that will best serve your needs.

An updated edition of the best tips and tools to plan, build, and execute a structured test operation In this update of his bestselling book, Rex Black walks you through how to develop essential tools and apply them to your test project. He helps you master the basic tools, apply the techniques to manage your resources, and give each area just the right amount of attention so that you can successfully survive managing a test project! Offering a thorough review of the tools and resources you will need to manage both large and small projects for hardware and software, this book prepares you to adapt the concepts across a broad range of settings. Simple and effective, the tools comply with industry standards and bring you up to date with the best test management practices and tools of leading hardware and software vendors. Rex Black draws from his own numerous testing experiences-- including the bad ones, so you can learn from his mistakes-- to provide you with insightful tips in test project management. He explores such topics as: Dates, budgets, and quality-expectations versus reality Fitting the testing process into the overall development or maintenance process How to choose and when to use test engineers and technicians, contractors and consultants, and external test labs and vendors Setting up and using an effective and simple bug-tracking database Following the status of each test case The companion Web site contains fifty tools, templates, and case studies that will help you put these ideas into action--fast!

Managing risk is essential for every organization. However, significant opportunities may be lost by concentrating on the negative aspects of risk without bearing in mind the positive attributes. The objective of Project Risk Management: Managing Software Development Risk is to provide a distinct approach to a broad range of risks and rewards associated with the design, development, implementation and deployment of software systems. The traditional perspective of software development risk is to view risk as a negative characteristic associated with the impact of potential threats. The perspective

of this book is to explore a more discerning view of software development risks, including the positive aspects of risk associated with potential beneficial opportunities. A balanced approach requires that software project managers approach negative risks with a view to reduce the likelihood and impact on a software project, and approach positive risks with a view to increase the likelihood of exploiting opportunities. Project Risk Management: Managing Software Development Risk explores software development risk both from a technological and business perspective. Issues regarding strategies for software development are discussed and topics including risks related to technical performance, outsourcing, cybersecurity, scheduling, quality, costs, opportunities and competition are presented. Bringing together concepts across the broad spectrum of software engineering with a project management perspective, this volume represents both a professional and scholarly perspective on the topic.

Formula for Success
Practical Tools and Techniques for Managing Hardware and Software Testing
Managing Iterative Software Development Projects
Software Project Management
Lessons Learned from Programming Over Time
Managing Software Development Projects
Statistical Process Control for Software Process Improvement

Managing Software Deliverables describes a set of proven processes for establishing an effective Software Program Management Office (SPMO) function in a corporate setting. Every business that has people performing Software Project Management (PM) activities has a need for these processes. In some instances, an organization may already have a PMO chartered with overall responsibility for each project managed in an enterprise. In those cases, this book will either provide validation of their efforts or it will provide some techniques and useful approaches that can be utilized to further improve on their overall implementation of the PMO. In the vast majority of cases in business, however, a PMO is unheard of. Each project managed in the enterprise is unfortunately managed separately from all others (at great cost in both time and money to the enterprise). Phase Roadmaps clearly depict what is expected by all parties at each phase of effort Pre-built Intranet is ready to deploy in a corporate setting and provides immediate use Process Methodology adheres to proven best-practices for software development

This book combines theory with practical knowledge of how Trac and Subversion can help us to achieve the process of managing the software development. This book is aimed at developers of all calibres, but particularly those that lead teams or projects, especially if they have recently moved into the role or are simply looking for a better way.

"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.

Recent growth in knowledge management concepts has played a vital role in the improvement of organizational performance. These knowledge management approaches have been influential in achieving the goal of efficient production of software development processes. Knowledge-Based Processes in Software Development focuses on the inherent issues to help practitioners in gaining understanding of software development processes. The best practices highlighted in this publication will be essential to software professionals working in the industry as well as students and researchers in the domain of software engineering in order to successfully employ knowledge management procedures.

Knowledge-Based Processes in Software Development
Managing Software Debt
Engineering and Managing Software Requirements
Managing Stakeholders in Software Development Projects
Adaptive Software Development
Building Great Software Engineering Teams
Rules, Tools, and Insights for Managing Software People and Teams

The concept of processes is at the heart of software and systems engineering. Software process models integrate software engineering methods and techniques and are the basis for managing large-scale software and IT projects. High product quality routinely results from high process quality. Software process management deals with getting and maintaining control over processes and their evolution. Becoming acquainted with existing software process models is not enough, though. It is important to understand how to select, define, manage, deploy, evaluate, and systematically evolve software process models so that they suitably address the problems, applications, and environments to which they are applied. Providing basic knowledge for these important tasks is the main goal of this textbook. Münch and his co-authors aim at providing knowledge that enables readers to develop useful process

models that are suitable for their own purposes. They start with the basic concepts. Subsequently, existing representative process models are introduced, followed by a description of how to create individual models and the necessary means for doing so (i.e., notations and tools). Lastly, different possible usage scenarios for process management are highlighted (e.g. process improvement and software process simulation). Their book is aimed at students and researchers working on software project management, software quality assurance, and software measurement; and at practitioners who are interested in process definition and management for developing, maintaining, and operating software-intensive systems and services.

"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.

WINNER of Computing Reviews 20th Annual Best Review in the category Management "Tyler's book is concise, reasonable, and full of interesting practices, including some curious ones you might consider adopting yourself if you become a software engineering manager." —Fernando Berzal, CR, 10/23/2015 "Josh Tyler crafts a concise, no-nonsense, intensely focused guide for building the workhouse of Silicon Valley—the high-functioning software team." —Gordon Rios, Summer Book Recommendations from the Smartest People We Know—Summer 2016 Building Great Software Engineering Teams provides engineering leaders, startup founders, and CTOs concrete, industry-proven guidance and techniques for recruiting, hiring, and managing software engineers in a fast-paced, competitive environment. With so much at stake, the challenge of scaling up a team can be intimidating. Engineering leaders in growing companies of all sizes need to know how to find great candidates, create effective interviewing and hiring processes, bring out the best in people and their work, provide meaningful career development, learn to spot warning signs in their team, and manage their people for long-term success. Author Josh Tyler has spent nearly a decade building teams in high-growth startups, experimenting with every aspect of the task to see what works best. He draws on this experience to outline specific, detailed solutions augmented by instructive stories from his own experience. In this book you'll learn how to build your team, starting with your first hire and continuing through the stages of development as you manage your team for growth and success. Organized to cover each step of the process in the order you'll likely face them, and highlighted by stories of success and failure, it provides an easy-to-understand recipe for creating your high-powered engineering team.

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 Americaâ__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.

Software Process Improvement and Management: Approaches and Tools for Practical Development

Recruiting, Hiring, and Managing Your Team from Startup to Success

Managing Software Development Risk

Process Improvement and CMMI for Systems and Software

Guide to Software Development

A Software Development Management Methodology

Reducing Friction in Software Development

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

Managing Humans is a selection of the best essays from Michael Lopp's web site, Rands in Repose. Drawing on Lopp's management experiences at Apple, Netscape, Symantec, and Borland, this book is full of stories based on companies in the Silicon Valley where people have been known to yell at each other. It is a place full of dysfunctional bright people who are in an incredible hurry to find the next big thing so they can strike it rich and then do it all over again. Among these people are managers, a strange breed of people who through a mystical organizational ritual have been given power over your future and your bank account. Whether you're an aspiring manager, a current manager, or just wondering what the heck a manager does all day, there is a story in this book that will speak to you. What you'll learn What to do when people start yelling at each other How to perform a diving save when the best engineer insists on resigning How to say "no" to the person who signs your paycheck Who this book is for This book is designed for managers and would-be managers staring at the role of a manager wondering why they would ever leave the safe world of bits and bites for the messy world of managing humans. The book covers handling conflict, managing wildly differing personality types, infusing innovation into insane product schedules, and figuring out how to build a lasting and useful engineering culture. Table of Contents Don't Be a Prick Managers are Not Evil The Monday Freakout Agenda Detection Mandate Dissection Information Starvation Subtlety, Subterfuge, and Silence Managementese Technicality

Avoiding the Fez Your Resignation Checklist Saying No 1.0 Taking Time to Think The Soak Malcolm Events Capturing Context Status Reports 2.0 Trickle Theory A Glimpse and a Hook Nailing the Phone Screen Ninety Days Bellwethers NADD A Nerd in a Cave Meeting Creatures Incrementalists and Completionists Organics and Mechanics Inwards, Outwards, and Holistics Free Electrons Rules for the Reorg Offshore Risk Factor Joe Secret Titles

Software development is a complex problem-solving activity with a high level of uncertainty. There are many technical challenges concerning scheduling, cost estimation, reliability, performance, etc, which are further aggravated by weaknesses such as changing requirements, team dynamics, and high staff turnover. Thus the management of knowledge and experience is a key means of systematic software development and process improvement. "Managing Software Engineering Knowledge" illustrates several theoretical examples of this vision and solutions applied to industrial practice. It is structured in four parts addressing the motives for knowledge management, the concepts and models used in knowledge management for software engineering, their application to software engineering, and practical guidelines for managing software engineering knowledge. This book provides a comprehensive overview of the state of the art and best practice in knowledge management applied to software engineering. While researchers and graduate students will benefit from the interdisciplinary approach leading to basic frameworks and methodologies, professional software developers and project managers will also profit from industrial experience reports and practical guidelines.

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.

Managing Complexity in Software Engineering

Project Risk Management

Innovation, Teamwork, and the Software Process

Software Processes and Life Cycle Models

How to Manage Your Software Projects, Your Teams, Your Boss, and Yourself

Managing Software Requirements

Managing Software Deliverables

Process Improvement and CMMI for Systems and Software provides a workable approach for achieving cost-effective improvements for systems and software. Focusing on planning, implementation, and management in system and softw processes, it supplies a brief overview of basic strategic planning models and covers fundamental concepts and appr This book focuses on the design, development, management, governance and application of evolving software process aligned with changing business objectives, such as expansion to new domains or shifting to global production. In the an evolving business world, it examines the complete software process lifecycle, from the initial definition of a produ systematic improvement. In doing so, it addresses difficult problems, such as how to implement processes in highly r domains or where to find a suitable notation system for documenting processes, and provides essential insights and readers manage process evolutions. And last but not least, it provides a wealth of examples and cases on how to dea evolution in practice. Reflecting these topics, the book is divided into three parts. Part 1 focuses on software busine transformation and addresses the questions of which process(es) to use and adapt, and how to organize process imp programs. Subsequently, Part 2 mainly addresses process modeling. Lastly, Part 3 collects concrete approaches, expe recommendations that can help to improve software processes, with a particular focus on specific lifecycle phases. T aimed at anyone interested in understanding and optimizing software development tasks at their organization. While t experiences and ideas presented will be useful for both those readers who are unfamiliar with software process impr want to get an overview of the different aspects of the topic, and for those who are experts with many years of ex particularly targets the needs of researchers and Ph.D. students in the area of software and systems engineering or systems who study advanced topics concerning the organization and management of (software development) project improvements projects.

"This is an incredibly wise and useful book. The authors have considerable real-world experience in delivering quality s that matter, and their expertise shines through in these pages. Here you will learn what technical debt is, what is it manage it, and how to pay it down in responsible ways. This is a book I wish I had when I was just beginning my care authors present a myriad of case studies, born from years of experience, and offer a multitude of actionable insights apply it to your project." –Grady Booch, IBM Fellow Master Best Practices for Managing Technical Debt to Promote S Quality and Productivity As software systems mature, earlier design or code decisions made in the context of budget constraints increasingly impede evolution and innovation. This phenomenon is called technical debt, and practical solut In Managing Technical Debt, three leading experts introduce integrated, empirically developed principles and practices software professional can use to gain control of technical debt in any software system. Using real-life examples, the explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and "paying off time. They describe how to establish managing technical debt as a core software engineering practice in your organiza Discover how technical debt damages manageability, quality, productivity, and morale–and what you can do about it C causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure Ide technical debt items, and analyze their costs so you can prioritize action Choose the right solution for each technical eliminate, reduce, or mitigate Integrate software engineering practices that minimize new debt Managing Technical D valuable resource for every software professional who wants to accelerate innovation in existing systems, or build ne that will be easier to maintain and evolve.

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

Reflections on Management

Guidelines for Improving the Software Process

Measuring the Software Process

Designing and Managing the Life Cycle

Strategies for Managing Computer Software Upgrades

Managing Humans
Managing the Testing Process

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.

Requirements engineering is the process by which the requirements for software systems are gathered, analyzed, documented, and managed throughout their complete lifecycle. Traditionally it has been concerned with technical goals for, functions of, and constraints on software systems. Aurum and Wohlin, however, argue that it is no longer appropriate for software systems professionals to focus only on functional and non-functional aspects of the intended system and to somehow assume that organizational context and needs are outside their remit. Instead, they call for a broader perspective in order to gain a better understanding of the interdependencies between enterprise stakeholders, processes, and software systems, which would in turn give rise to more appropriate techniques and higher-quality systems. Following an introductory chapter that provides an exploration of key issues in requirements engineering, the book is organized in three parts. Part 1 presents surveys of state-of-the art requirements engineering process research along with critical assessments of existing models, frameworks and techniques. Part 2 addresses key areas in requirements engineering, such as market-driven requirements engineering, goal modeling, requirements ambiguity, and others. Part 3 concludes the book with articles that present empirical evidence and experiences from practices in industrial projects. Its broader perspective gives this book its distinct appeal and makes it of interest to both researchers and practitioners, not only in software engineering but also in other disciplines such as business process engineering and management science.

Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world's leading practitioners construct and maintain software. This book covers Google's unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You'll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

Part of the new Digital Filmmaker Series! Digital Filmmaking: An Introductionis the first book in the newDigital Filmmaker Series. Designed for an introductory level course in digital filmmaking, it is intended for anyone who has an interest in telling stories with pictures and sound and won't assume any familiarity with equipment or concepts on the part of the student. In addition to the basics of shooting and editing, different story forms are introduced from documentary and live events through fictional narratives. Each of the topics is covered in enough depth to allow anyone with a camera and a computer to begin creating visual projects of quality.

Software Process Modeling
Software Engineering at Google
The Capability Maturity Model
Managing Software Engineering Knowledge
A Collaborative Approach to Managing Complex Systems
Managing the Development of Software-Intensive Systems
A Unified Framework

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, deaflines and budgets.

"The speed with which companies are bringing new software products to market is having a serious impact on information technology use in organizations. As vendors release new software products, customers are faced with the prospect of upgrading to the new software. If not managed properly, the upgrade might cost inordinate amounts of money and/or curtail employee productivity. To aid IT managers, this book provides strategies for managing issues associated with the

implementation of software upgrades. In addition, the book presents selected research papers which provide indepth treatment of the most critical aspects of software upgrade management"--Provided by publisher.

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 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

A Unified Approach
Software Process Definition and Management
Managing Software Development with Trac and Subversion
Traditional, Agile and Beyond - How to Handle Process Change
Managing the Software Process
Managing the Unmanageable
An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes

This book addresses how best to make build vs. buy decisions, and what effect such decisions have on the software development life cycle (SDLC). Offering an integrated approach that includes important management and decision practices, the text explains how to create successful solutions that fit user and customer needs, by mixing different SDLC methodologies. Features: provides concrete examples and effective case studies; focuses on the skills and insights that distinguish successful software implementations; covers management issues as well as technical considerations, including how to deal with political and cultural realities in organizations; identifies many new alternatives for how to manage and model a system using sophisticated analysis tools and advanced management practices; emphasizes how and when professionals can best apply these tools and practices, and what benefits can be derived from their application; discusses searching for vendor solutions, and vendor contract considerations.

As stakeholder relationships and business in general have become increasingly central to the unfolding of stakeholder thinking, important new topics have begun to take centre stage in both the worlds of practitioners and academics. The role of project management becomes immeasurably more challenging, when stakeholders are no longer seen as simple objects of managerial action but rather as subjects with their own objectives and purposes. This book will aim to explain some of the complexities of project management and managerial relationships with stakeholders by discussing the practice of stakeholder engagement, dialog, measurement and management and the consequences of this practice for reporting and productivity, and performance within project management.

The Practical, Start-to-Finish Guide to Planning and Leading Iterative Software Projects Iterative processes have gained widespread acceptance because they help software developers reduce risk and cost, manage change, improve productivity, and deliver more effective, timely solutions. But conventional project management techniques don[]t work well in iterative projects, and newer iterative management techniques have been poorly documented. Managing Iterative Software Development Projects is the solution: a relentlessly practical guide to planning, organizing, estimating, staffing, and managing any iterative project, from start to finish. Leading iterative development experts Kurt Bittner and Ian Spence introduce a proven, scalable approach that improves both agility and control at the same time, satisfying the needs of developers, managers, and the business alike. Their techniques are easy to understand, and easy to use with any iterative methodology, from Rational Unified Process to Extreme Programming to the Microsoft Solutions Framework. Whatever your role[]team leader, program manager, project manager, developer, sponsor, or user representative[]this book will help you Understand the key drivers of success in iterative projects Leverage []time boxing[] to define project lifecycles and measure results Use Unified Process phases to facilitate controlled iterative development Master core concepts of iterative project management, including layering and evolution Create project roadmaps, including release plans Discover key patterns of risk management, estimation, organization, and iteration planning Understand what must be controlled centrally, and what you can safely delegate Transition smoothly to iterative processes Scale iterative project management from the smallest to the largest projects Align software investments with the needs of the business Whether you are interested in software development using RUP, OpenUP, or other agile processes, this book will help you reduce the anxiety and cost associated with software improvement by providing an easy, non-intrusive path toward improved results[]without overwhelming you and your team.

This book covers complex software engineering projects, new paradigms for system development, object-orientated design and formal methods, project management and automation perspectives.

Managing Software Process Evolution
Managing and Leading Software Projects
Introduction to the Team Software Process(sm)
Building for Inevitable Change (Adobe Reader)
Managing Technical Debt

This thoroughly updated edition of the bestselling original is short on theory and long on practical, hands-on advice. Packed with realistic scenarios and case studies, this book shows readers how to avoid most of the problems encountered in the software development process.

Over the past decade, there has been an increase in attention and focus on the discipline of software engineering. Software engineering tools and techniques have been developed to gain more predictable quality improvement results. Process standards such as Capability Maturity Model Integration (CMMI), ISO 9000, Software Process Improvement and Capability dEtermination (SPICE), Agile Methodologies, and others have been proposed to assist organizations to achieve more predictable results by incorporating these proven standards and procedures into their software process. Software Process Improvement and Management: Approaches and Tools for Practical Development offers the latest research and case studies on software engineering and development. The production of new process standards assist organizations and software engineers in adding a measure of predictability to the software process. Companies can gain a decisive competitive advantage by applying these new and theoretical methodologies in real-world scenarios. Researchers, scholars, practitioners, students, and anyone interested in the field of software development and design should access this book as a major compendium of the latest research in the field.
This book provides a comprehensive overview of the field of software processes, covering in particular the following essential topics: software process modelling, software process and lifecycle models, software process management, deployment and governance, and software process improvement (including assessment and measurement). It does not propose any new processes or methods; rather, it introduces students and software engineers to software processes and life cycle models, covering the different types ranging from "classical", plan-driven via hybrid to agile approaches. The book is structured as follows: In chapter 1, the fundamentals of the topic are introduced: the basic concepts, a historical overview, and the terminology used. Next, chapter 2 covers the various approaches to modelling software processes and lifecycle models, before chapter 3 discusses the contents of these models, addressing plan-driven, agile and hybrid approaches. The following three chapters address various aspects of using software processes and lifecycle models within organisations, and consider the management of these processes, their assessment and improvement, and the measurement of both software and software processes. Working with software processes normally involves various tools, which are the focus of chapter 7, before a look at current trends in software processes in chapter 8 rounds out the book. This book is mainly intended for graduate students and practicing professionals. It can be used as a textbook for courses and lectures, for self-study, and as a reference guide. When used as a textbook, it may support courses and lectures on software processes, or be used as complementary literature for more basic courses, such as introductory courses on software engineering or project management. To this end, it includes a wealth of examples and case studies, and each chapter is complemented by exercises that help readers gain a better command of the concepts discussed.