Practical Software Development Techniques 3rd Edition Tools And Techniques For Building Enterprise

Designing Software ArchitecturesUser Stories AppliedSoftware EngineeringProceedings of the 3rd International Workshop on Software Configuration ManagementProceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development EnvironmentsCMMI for DevelopmentEvaluation of Novel Approaches to Software EngineeringSoftware EngineeringSoftware Engineering and Formal MethodsDesigning Software-Intensive Systems: Methods and PrinciplesA Practical Handbook for Software DevelopmentPractical Software Development TechniquesThe Art of Software TestingBeginning Software EngineeringPractical Enterprise Software Development TechniquesVersion Control with GitThird International Conference on the Software ProcessFUNDAMENTALS OF SOFTWARE ENGINEERINGSoftware MetricsEncyclopedia of Information Ethics and SecurityPractical Software Project Estimation: A Toolkit for Estimating Software Development Effort & DurationESEC '911993 Software Engineering Environments Conference, July 7-9, 1993, Reading, United KingdomRequirements Engineering for Software and Systems, Second EditionManaging the Testing ProcessHands-On Application Development with PyCharmACM Transactions on Software Engineering and MethodologyMining

Software Engineering Data for Software ReuseBusinessTutorial--software Engineering Project ManagementSoftware Engineering EnvironmentsSoftware Process Improvement for Small and Medium Enterprises: Techniques and Case Studies3rd USA-Japan Computer Conference ProceedingsCode CompleteThe Rational Unified ProcessSoftware MetricsObject-oriented Software EngineeringCreating a Software Engineering CultureFundamentals of Software EngineeringStructured Object-Oriented Formal Language and Method

Designing Software Architectures

Software engineering is of major importance to all enterprises; however, the key areas of software quality and software process improvement standards and models are currently geared toward large organizations, where most software organizations are small and medium enterprises. Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies offers practical and useful guidelines, models, and techniques for improving software processes and products for small and medium enterprises, utilizing the authoritative, demonstrative tools of case studies and lessons learned to provide academics, scholars, and practitioners with an invaluable research source.

User Stories Applied

Software Engineering

Reprints and five new papers present a top-down view of the subject. Covers software engineering and SE project management planning, organizing, staffing, directing, and controlling a SE project. No index. Annotation copyright Book News, Inc. Portland, Or.

Proceedings of the 3rd International Workshop on Software Configuration Management

Annotation Proceedings of the Software Engineering Environments Conference, July 7-9, 1993, Reading, UK. It took up principles and models, architecture and technology, experiments and experiences in the field. No index. Annotation copyright by Book News, Inc., Portland, OR.

Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments

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

CMMI for Development

Widely considered one of the best practical guides to programming, Steve McConnell’s original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help you: Design for minimum complexity and maximum creativity Reap the benefits of collaborative development Apply defensive programming techniques to reduce and flush out errors Exploit opportunities to refactor—or evolve—code, and do it safely Use construction practices that are right-weight for your project Debug problems quickly and effectively Resolve critical construction issues early and correctly Build quality into the beginning, middle, and end of your project

Evaluation of Novel Approaches to Software Engineering

This book provides an overview of tools and techniques used in enterprise software development, many of which are not taught in academic programs or learned on the job. This is an ideal resource containing lots of practical information and code examples that you need to master as a member of an enterprise development team. This book aggregates many of these "on the job" tools and techniques into a concise format and presents them as both discussion topics and with code examples. The reader will not only get an overview of these tools and techniques, but also several discussions concerning operational aspects of enterprise software development and how it differs from smaller development efforts. For example, in the chapter on Design Patterns and Architecture, the author describes the basics of design patterns but only highlights those that are more important in enterprise applications due to separation of duties, enterprise security, etc. The architecture discussion revolves has a similar emphasis – different teams may manage different aspects of the application’s components with little or no access to the developer. This aspect of restricted access is also mentioned in the section on logging. Theory of logging and discussions of what to log are briefly mentioned, the configuration of the logging tools is demonstrated along with a discussion of why it’s very important in an enterprise environment.

Software Engineering

Rapid technological advancement has given rise to new ethical dilemmas and security threats, while the development of appropriate ethical codes and security measures fail to keep pace, which makes the education of computer users and professionals crucial. The Encyclopedia of Information Ethics and Security is an original, comprehensive reference source on ethical and security issues relating to the latest technologies. Covering a wide range of themes, this valuable reference tool includes topics such as computer crime, information warfare, privacy, surveillance, intellectual property and education. This encyclopedia is a useful tool for students, academics, and professionals.

Software Engineering and Formal Methods

Thoroughly reviewed and eagerly anticipated by the agile community, User Stories Applied offers a requirements process that saves time, eliminates rework, and leads directly to better software. The best way to build software that meets users' needs is to begin with "user stories": simple, clear, brief descriptions of functionality that will be valuable to real users. In User Stories Applied, Mike Cohn provides you with a front-to-back blueprint for writing these user stories and weaving them into your development lifecycle. You'll learn what makes a great user story, and what makes a bad one. You'll discover practical ways to gather user stories, even when you can't speak with your users. Then, once you've compiled your user stories, Cohn shows how to organize them, prioritize them, and use them for planning, management, and testing. User role modeling: understanding what users have in common, and where they differ Gathering stories: user interviewing, questionnaires, observation, and workshops Working with managers, trainers, salespeople and other "proxies" Writing user stories for acceptance testing Using stories to prioritize, set schedules, and estimate release costs Includes end-of-chapter practice questions and exercises User Stories Applied will be invaluable to every software developer, tester, analyst, and manager working with any agile method: XP, Scrum or even your own home-grown approach.

Designing Software-Intensive Systems: Methods and Principles

This monograph discusses software reuse and how it can be applied at different stages of the software development process, on different types of data and at different levels of granularity. Several challenging hypotheses are analyzed and confronted using novel data-driven methodologies, in order to solve problems in requirements elicitation and specification extraction, software design and implementation, as well as software quality assurance. The book is accompanied by a number of tools, libraries and working prototypes in order to practically illustrate how the phases of the software engineering life cycle can benefit from unlocking the potential of data. Software engineering researchers, experts, and practitioners can benefit from the various methodologies presented and can better understand how knowledge extracted from software data residing in various repositories can be combined and used to enable effective decision making and save considerable time and effort through software reuse. Mining Software Engineering Data for Software Reuse can also prove handy for graduate-level students in software engineering.

A Practical Handbook for Software Development

bull; Reflects all of the changes that were integrated into RUP v2003-the latest version of the very popular produc t bull; Learn the key concepts, fundamentals of structure, integral content, and motivation behind the RUP bull; Covers all phases of the software development lifecycle -from concept, to delivery, to revision

Practical Software Development Techniques

This volume presents the proceedings of the Third European Software Engineering Conference. Themes include formal methods and practical experiences with them, special techniques for real-time systems, software evolution and re-engineering, software engineering environments, and software metrics.

The Art of Software Testing

Beginning Software Engineering

Get up to speed on Git for tracking, branching, merging, and managing code revisions. Through a series of step-by-step tutorials, this practical guide takes you quickly from Git fundamentals to advanced techniques, and provides friendly yet rigorous advice for navigating the many functions of this open source version control system. This thoroughly revised edition also includes tips for manipulating trees, extended coverage of the reflog and stash, and a complete introduction to the GitHub repository. Git lets you manage code development in a virtually endless variety of ways, once you understand how to harness the system’s flexibility. This book shows you how. Learn how to use Git for several real-world development scenarios Gain insight into Git’s common-use cases, initial tasks, and basic functions Use the system for both centralized and distributed version control Learn how to manage merges, conflicts, patches, and diffs Apply advanced techniques such as rebasing, hooks, and ways to handle submodules Interact with Subversion (SVN) repositories—including SVN to Git conversions Navigate, use, and contribute to open source projects though GitHub

Practical Enterprise Software Development Techniques

This is the most authoritative archive of Barry Boehm's contributions to software engineering. Featuring 42 reprinted articles, along with an introduction and chapter summaries to provide context, it serves as a "how-to" reference manual for software engineering best practices. It provides convenient access to Boehm's landmark work on product development and management processes. The book concludes with an insightful look to the future by Dr. Boehm.

Version Control with Git

This book constitutes the refereed proceedings of the 10th International Conference on Software Engineering and Formal Methods, SEFM 2012, held in Thessaloniki, Greece, in October 2012. The 19 revised research papers presented together with 3 short papers, 2 tool papers, and 2 invited talks were carefully reviewed and selected from 98 full submissions. The SEFM conference aspires to advance the state-of-the-art in formal methods, to enhance their scalability and usability with regards to their application in the software industry and to promote their integration with practical engineering methods.

Third International Conference on the Software Process

This book covers the essential knowledge and skills needed by a student who is specializing in software engineering. Readers will learn principles of object orientation, software development, software modeling, software design, requirements analysis, and testing. The use of the Unified Modelling Language to develop software is taught in depth. Many concepts are illustrated using complete examples, with code written in Java.

FUNDAMENTALS OF SOFTWARE ENGINEERING

Software Metrics

□□□□□□□□□

Encyclopedia of Information Ethics and Security

Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort & Duration

The present volume contains the proceedings of the Third IPM International

Conference on Fundamentals of Software Engineering (FSEN), Kish, Iran, April 15–17, 2009. FSEN 2009 was organized by the School of Computer Science at the Institute for Studies in Fundamental Sciences (IPM) in Iran, in cooperation with the ACM SIGSOFT and IFIP WG 2.2. This conference brought together around 100 researchers and practitioners working on di?erent aspects of formal methods in software engineering from 15 di?erentcountries.ThetopicsofinterestinFSENspanoverallaspects offormal methods,especiallythoserelatedtoadvancingtheapplicationofformalmethods in software industry and promoting their integration with practical engineering techniques. The Program Committee of FSEN 2009 consisted of top researchers from 24 di?erent academic institutes in 11 countries. We received a total of 88 submissions from 25 countries out of which the Program Committee selected 22 as regular papers, 5 as short papers, and 7 as poster presentations in the conferenceprogram.Eachsubmissionwasreviewedbyatleastthreeindependent referees, for its quality, originality, contribution, clarity of presentation, and its relevance to the conference topics. This volume contains the revised versions of the regular and short papers presented at FSEN 2009. Three distinguished keynote speakers delivered their lectures at FSEN 2009 on models of computation: automata and processes (Jos Baeten), veri?cation, performanceanalysisandcontrollersynthesisforreal-timesystems(KimLarsen), and theory and tool for component-based model-driven development in rCOS (Zhiming Liu). Our invited speakers also contributed to this volume by s- mitting their keynote papers, which were accepted after they were reviewed by independent referees.

ESEC '91

As requirements engineering continues to be recognized as the key to on-time and on-budget delivery of software and systems projects, many engineering programs have made requirements engineering mandatory in their curriculum. In addition, the wealth of new software tools that have recently emerged is empowering practicing engineers to improve their requirements engineering habits. However, these tools are not easy to use without appropriate training. Filling this need, Requirements Engineering for Software and Systems, Second Edition has been vastly updated and expanded to include about 30 percent new material. In addition to new exercises and updated references in every chapter, this edition updates all chapters with the latest applied research and industry practices. It also presents new material derived from the experiences of professors who have used the text in their classrooms. Improvements to this edition include: An expanded introductory chapter with extensive discussions on requirements analysis, agreement, and consolidation An expanded chapter on requirements engineering for Agile methodologies An expanded chapter on formal methods with new examples An expanded section on requirements traceability An updated and expanded section on requirements engineering tools New exercises including ones suitable for research projects Following in the footsteps of its bestselling predecessor, the text illustrates key ideas associated with requirements engineering using extensive case studies and three common example systems: an airline baggage handling system, a point-of-sale system for a large pet store chain, and a system for a smart home. This edition also includes an example of a wet well pumping system for a wastewater treatment station. With a focus on software-intensive systems, but highly applicable to non-software systems, this text provides a probing and comprehensive review of recent developments in requirements engineering in high integrity systems.

1993 Software Engineering Environments Conference, July 7-9, 1993, Reading, United Kingdom

This book constitutes revised selected papers from the Third International Workshop on Structured Object-Oriented Formal Language and Method, SOFL+MSVL 2013, held in Queenstown, New Zealand, in October 2013. The 13 papers presented in this volume were carefully reviewed and selected from 22 submissions. They are organized in topical sections on testing and verification, simulation and model checking, SOFL tools, and formal specification and application.

Requirements Engineering for Software and Systems, Second Edition

Product verifiable, defensible, and achievable software estimates Based on data collected by the International Software Benchmarking Standards Group (ISBSG), Practical Software Project Estimation explains how to accurately forecast the size, cost, and schedule of software projects. Get expert advice on generating accurate estimates, minimizing risks, and planning and managing projects. Valuable appendixes provide estimation equations, delivery rate tables, and the ISBSG Repository demographics. Verify project objectives and requirements Determine, validate, and refine software functional size Produce indicative estimates using regression equations Predict effect and duration through comparison and analogy Build estimation frameworks Perform benchmarks using the ISBSG Repository Compare IFPUG, COSMIC, and FiSMA sizing methods Peter Hill is the chief executive officer and a director of the ISBSG. He has been in the information services industry for more than 40 years and has compiled and edited five books for the ISBSG.

Managing the Testing Process

PART I: FUNDAMENTALS OF MEASUREMENT AND EXPERIMENTATION 1.

Measurement: What Is It and Why Do It? 2. The Basics of Measurement 3. A Goal-Based Framework for Software Measurement 4. Empirical Investigation 5. Software Metrics Data Collection 6. Analyzing Software-Measurement Data PART II: SOFTWARE-ENGINEERING MEASUREMENT 7. Measuring Internal Product Attributes: Size 8. Measuring Internal Product Attributes: Structure 9. Measuring Internal Product Attributes 10. Software Reliability: Measurement and Prediction 11. Resource Measurement: Productivity, Teams, and Tools 12. Making Process Predictions PART III: MEASUREMENT AND MANAGEMENT 13. Planning a Measurement Program 14. Measurement in Practice 15. Empirical Research in Software Engineering APPENDIXES: A. Solutions to Selected Exercises / B. Metric Tools / C. Acronyms and Glossary / ANNOTATED BIBLIOGRAPHY / INDEX

Hands-On Application Development with PyCharm

A guide to the development process covers phase planning, indicators, models, configuration, project inception, system definition, design, and production, and project debriefing

ACM Transactions on Software Engineering and Methodology

Designing Software Architectures is the first step-by-step guide to making the crucial design decisions that can make or break your software architecture. SEI expert Rick Kazman and Dr. Humberto Cervantes provide comprehensive guidance for ensuring that your architectural design decisions are consistently rational and evidence-based. Drawing on their own extensive experience, they demonstrate how to craft designs that are practical and effective, and support all phases of architectural development, from requirements to documentation. You'll learn how to successfully integrate the design process in an organizational context, including designing systems that will be built with agile methods. The authors begin with a general review of software architecture concepts and the software architecture lifecycle. Next, they explain what architecture design really means, introduce key design concepts and principles, and walk through both conventional and alternative design processes. Building on this foundation, they introduce the new Attribute-Driven Design (ADD) 3.0 process, walk the reader through two extended ADD 3.0 case studies, and demonstrate how ADD 3.0 can lead to more successful designs. You'll learn how to scale design and analysis up and down - for example, to design for pre-sales processes and lightweight architecture reviews. Kazman and Cervantes illuminate the relationships between analysis and design, introduce a set of reusable design primitives, and identify issues and solutions for new domains, including cloud, mobile, and big data. Design is the core activity for software designers and architects, but for most practitioners, it's been a black art. This book offers the systematic guidance you need to consistently do it rationally, and do it right.

Mining Software Engineering Data for Software Reuse

Featuring an associated Web page, and consistently combining theory with real-world practical applications, this text includes thought-provoking questions about legal and ethical issues in software engineering.

Business

A complete introduction to building robust and reliable software Beginning Software Engineering demystifies the software engineering methodologies and techniques that professional developers use to design and build robust, efficient, and consistently reliable software. Free of jargon and assuming no previous programming, development, or management experience, this accessible guide explains important concepts and techniques that can be applied to any programming language. Each chapter ends with exercises that let you test your understanding and help you elaborate on the chapter's main concepts. Everything you need to understand waterfall, Sashimi, agile, RAD, Scrum, Kanban, Extreme Programming, and many other development models is inside! Describes in plain English what software engineering is Explains the roles and responsibilities of team members working on a software engineering project Outlines key phases that any software engineering effort must handle to produce applications that are powerful and dependable Details the most popular software development methodologies and explains the different ways they handle critical development tasks Incorporates exercises that expand upon each chapter's main ideas Includes an extensive glossary of software engineering terms

Tutorial--software Engineering Project Management

This expanded and updated edition of "Practical Enterprise Software Development Techniques" includes a new chapter which explains what makes enterprise scale software development different from other development endeavors. Chapter 4 has been expanded with additional coverage of code review, bug tracker systems and agile software applications. The chapter order has been changed in response to feedback from readers and instructors who have taught classes using the previous version (which was also published by Apress). This book provides an overview of tools and techniques used in enterprise software development, many of which are not taught in academic programs or learned on the job. This is an ideal resource containing lots of practical information and code examples that you need to master as a member of an enterprise development team. This book aggregates many of these "on the job" tools and techniques into a concise format and presents them as both discussion topics and with code examples. The reader will not only get an overview of these tools and techniques, but also several discussions concerning operational aspects of enterprise software development and how it differs from smaller development efforts. For example, in the chapter on Design Patterns and Architecture, the author describes the basics of design patterns but only highlights those that are more important in enterprise applications due to separation of duties, enterprise security, etc. The architecture discussion revolves has a similar emphasis – different teams may manage different aspects of the application's components with little or no access to the developer. This aspect of restricted access is also mentioned in the section on logging. Theory of logging and discussions of what to log are briefly mentioned, the configuration of the logging tools is demonstrated along with a discussion of why it's very important in an enterprise environment.

Software Engineering Environments

The classic, landmark work on software testing The hardware and software of computing have changed markedly in the three decades since the first edition of The Art of Software Testing, but this book's powerful underlying analysis has stood the test of time. Whereas most books on software testing target particular development techniques, languages, or testing methods, The Art of Software Testing, Third Edition provides a brief but powerful and comprehensive presentation of time-proven software testing approaches. If your software development project is mission critical, this book is an investment that will pay for itself with the first bug you find. The new Third Edition explains how to apply the book's classic principles to today's hot topics including: Testing apps for iPhones, iPads, BlackBerrys, Androids, and other mobile devices Collaborative (user) programming and testing Testing for Internet applications, e-commerce, and agile programming environments Whether you're a student looking for a testing guide you'll use for the rest of your career, or an IT manager overseeing a software development team, The Art of Software Testing, Third Edition is an expensive book that will pay for itself many times over.

Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies

This is the digital version of the printed book (Copyright © 1996). Written in a remarkably clear style, Creating a Software Engineering Culture presents a comprehensive approach to improving the quality and effectiveness of the software development process. In twenty chapters spread over six parts, Wiegers promotes the tactical changes required to support process improvement and high-quality software development. Throughout the text, Wiegers identifies scores of culture builders and culture killers, and he offers a wealth of references to resources for the software engineer, including seminars, conferences, publications, videos, and on-line information. With case studies on process improvement and software metrics programs and an entire part on action planning (called “What to

Do on Monday"), this practical book guides the reader in applying the concepts to real life. Topics include software culture concepts, team behaviors, the five dimensions of a software project, recognizing achievements, optimizing customer involvement, the project champion model, tools for sharing the vision, requirements traceability matrices, the capability maturity model, action planning, testing, inspections, metrics-based project estimation, the cost of quality, and much more! Principles from Part 1 Never let your boss or your customer talk you into doing a bad job. People need to feel the work they do is appreciated. Ongoing education is every team member's responsibility. Customer involvement is the most critical factor in software quality. Your greatest challenge is sharing the vision of the final product with the customer. Continual improvement of your software development process is both possible and essential. Written software development procedures can help build a shared culture of best practices. Quality is the top priority; long-term productivity is a natural consequence of high quality. Strive to have a peer, rather than a customer, find a defect. A key to software quality is to iterate many times on all development steps except coding: Do this once. Managing bug reports and change requests is essential to controlling quality and maintenance. If you measure what you do, you can learn to do it better. You can't change everything at once. Identify those changes that will yield the greatest benefits, and begin to implement them next Monday. Do what makes sense; don't resort to dogma.

3rd USA-Japan Computer Conference Proceedings

Advancements and rapid developments have led to many ramifications in the ever-changing world of software engineering. This book, in its third edition, is restructured and revised to trace the advancements made and landmarks achieved in the field. This book not only incorporates latest and enhanced software engineering techniques and practices but also shows how these techniques are applied into the practical software assignments. The chapters are incorporated with illustrative examples to add an analytical insight on the subject. The book is logically organized to cover expanded and revised treatment of all software process activities. New to This Edition : The contents and presentation of all chapters have been improved thoroughly. A new layout highlights the important concepts and theories in a box format, in order to help the students in learning process. A new chapter on Emerging Trends emphasizes on the most upcoming and advanced techniques in the field of Software Engineering. Primarily intended for the undergraduate students of Software Engineering, the book is also beneficial for the students opting for a course in MCA, MBA and IT. Key Features : Large number of worked-out examples and practice problems. Chapter-end exercises and solutions to selected problems to check students’ comprehension on the subject. Solutions manual available for instructors. Power Point slides available online at www.phindia.com/rajibmall to provide integrated learning to the students.

Code Complete

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.

The Rational Unified Process

Software engineering is understood as a broad term linking science, traditional enneering, art and management and is additionally conditioned by social and external factors (conditioned to the point that brilliant engineering solutions based on strong science, showing artistic creativity and skillfully managed can still fail for reasons beyond the control of the development team). Modern software engineering needs a paradigm shift commensurate with a change of the computing paradigm from: 1. Algorithms to interactions (and from procedural to object-oriented programming) 2. Systems development to systems integration 3.Products to services Traditional software engineering struggles to address this paradigm shift to inter- tions, integration, and services. It offers only incomplete and disconnected methods for building information systems with fragmentary ability to dynamically accom- date change and to grow gracefully. The principal objective of contemporary software engineering should therefore be to try to redefine the entire discipline and offer a complete set of methods, tools and techniques to address challenges ahead that will shape the information systems of the future.

Software Metrics

A definitive guide to PyCharm to help you build business-oriented Python applications ranging from modern web development to data science Key Features Learn basic to advanced PyCharm concepts to improve efficiency of your Python projects Work through practical examples that focus on efficient application development with PyCharm Explore advanced features in PyCharm such as code automation, version control, and GUI debugging Book Description JetBrain's PyCharm is the most popular Integrated Development Environment (IDE) used by the Python community thanks to its numerous features that facilitate faster, more accurate, and more productive programming practices. However, the abundance of options and customizations can make PyCharm seem quite intimidating. Hands-on Application Development with PyCharm starts with PyCharm's installation and configuration process, and systematically takes you through a number of its powerful features that can greatly improve your productivity. You'll explore code automation, version control, graphical debugging/testing, management of virtual environments, and much more. Finally, you'll delve into specific PyCharm features that support web development and data science, two of the fastest growing applications in Python programming. These include the integration of the Django framework as well as the extensive support for IPython and Jupyter Notebook. By the end of this PyCharm book, you will have gained extensive knowledge of the tool and be able to implement its features and make the most of its support for your projects. What you will learn Explore PyCharm functionalities and what makes it stand out from other Python IDEs Set up, configure, and customize your Python projects in PyCharm Understand how PyCharm integrates with Django for web development Discover PyCharm's capabilities in database management and data visualization Perform code automation, GUI testing, and version control in PyCharm Integrate interactive Python tools such as Jupyter Notebooks for building virtual environments Who this book is for If you’re a beginner or an expert Python user looking to improve your productivity using one of the best Python IDEs, this book is for you. Basic knowledge of Python programming language is expected.

Object-oriented Software Engineering

A Framework for Managing, Measuring, and Predicting Attributes of Software Development Products and Processes Reflecting the immense progress in the development and use of software metrics in the past decades, Software Metrics: A Rigorous and Practical Approach, Third Edition provides an up-to-date, accessible, and comprehensive introduction to software metrics. Like its popular predecessors, this third edition discusses important issues, explains essential concepts, and offers new approaches for tackling long-standing problems. New to the Third Edition This edition contains new material relevant to object-oriented design, design patterns, model-driven development, and agile development processes. It includes a new chapter on causal models and Bayesian networks and their application to software engineering. This edition also incorporates recent references to the latest software metrics activities, including research results, industrial case studies, and standards. Suitable for a Range of Readers With numerous examples and exercises, this book continues to serve a wide audience. It can be used as a textbook for a software metrics and quality assurance course or as a useful supplement in any software engineering course. Practitioners will appreciate the important results that have previously only appeared in research-oriented publications. Researchers will welcome the material on new results as well as the extensive bibliography of measurement-related information. The book also gives software managers and developers practical guidelines for selecting metrics and planning their use in a measurement program.

Creating a Software Engineering Culture

Fundamentals of Software Engineering

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!

Structured Object-Oriented Formal Language and Method

A remarkably compelling 86 minutes of BN and Montana Rail Link trains crowding this single track line between Spokane, WA and Sandpoint, ID. To watch all trains in the funnel for a 24 hour period would be deadly without the fine narrative (origins, destinations, cargo, history of the explosive growt

ROMANCE ACTION & ADVENTURE MYSTERY & THRILLER BIOGRAPHIES & HISTORY CHILDREN’S YOUNG ADULT FANTASY HISTORICAL FICTION HORROR LITERARY FICTION NON-FICTION SCIENCE FICTION