Fundamentals Of Software Engineering

Taking a learn-by-doing approach, Software Engineering Design: Theory and Practice uses examples, review questions, chapter exercises, and case study assignments to provide students and practitioners with the understanding required to design complex software systems. Explaining the concepts that are immediately relevant to software designers, it begins with a review of software design fundamentals. The text presents a formal top-down design process that consists of several design activities with varied levels of detail, including the macro-, micro-, and construction-design levels. As part of the top-down approach, it provides in-depth coverage of applied architectural, creational, structural, and behavioral design patterns. For each design issue covered, it includes a step-by-step breakdown of the execution of the design solution, along with an evaluation, discussion, and justification for using that particular solution. The book outlines industry-proven software design practices for leading large-scale software design efforts, developing reusable and high-quality software systems, and producing technical and customer-driven design documentation. It also: Offers one-stop guidance for mastering the Software Design & Construction sections of the official Software Engineering Body of Knowledge (SWEBOK®) Details a collection of standards and guidelines for structuring high-quality code Describes techniques for analyzing and evaluating the quality of software designs Collectively, the text supplies comprehensive coverage of the software design concepts students will need to succeed as professional design leaders. The section on engineering leadership for software designers covers the necessary ethical and leadership skills required of software developers in the public domain. The section on creating software design documents (SDD) familiarizes students with the software design notations, structural descriptions, and behavioral models required for SDDs. Course notes, exercises with answers, online resources, and an instructor's manual are available upon qualified course adoption. Instructors can contact the author about these resources via the author's website: http://softwareengineeringdesign.com/

Explore the latest Java-based software development techniques and methodologies through the project-based approach in this practical guide. Unlike books that use abstract examples and lots of theory, Real-World Software Development shows you how to develop several relevant projects while learning best practices along the way. With this engaging approach, junior developers capable of writing basic Java code will learn about state-of-the-art software development practices for building modern, robust and maintainable Java software. You'll work with many different software development topics that are often excluded from software develop how-to references. Featuring real-world examples, this book teaches you techniques and methodologies for functional programming, automated testing, security, architecture, and distributed systems.

The discipline of engineering which focuses on building robust software systems is termed as software engineering. The primary objective of software engineering is to create solutions which are able to meet their users' requirements. Software engineering is applied to small, medium and large-scale organizations. It utilizes engineering methods, processes, and techniques to create effective software solutions. According to the availability of resources, software development can be done by a team or an individual. Network control systems, operating systems, computer games and business applications are some common applications of software engineering. Software design, software development, software testing and software maintenance are few of its various sub-fields. Changing technology and new areas of specialization are evolving this field at a rapid pace. The topics included in this book on software engineering are of utmost significance and bound to provide incredible insights to readers. While understanding the long-term perspectives of the topics, it makes an effort in highlighting their impact as a modern tool for the growth of the discipline. For all those who are interested in software engineering, this book can prove to be an essential guide.

Want to venture into software engineering, but don't know where to begin? Now that technology has made its way to all industries, knowing how to wield its power has become a must-have skill. Yet although tech based competencies are a necessity, most people still hesitate to develop their skills, intimidated by the amount of material available. Software engineering is no exception. Many people think having a degree is an absolute must before you can become a software engineer. But that's simply not true. Kickstart your software engineering journey with How to Transition Into Software Engineering in 120 Days! Use this book as a guide for navigating the technicalities of software engineering. Tackle basic and advanced competencies in computer science and development. Unlike overly complicated books, ours aim to help beginners new to the field and concepts of software engineering, while also supplementing the knowledge base of experts and professionals. With our help, you can build your arsenal and equip yourself with tools you'll need for a career in software engineering--all in 120 days. Combine theoretical concepts and hone your craft with the help of our book's no-fuss and easy-to-understand approach. Learn how to solve problems, innovate solutions, and bring your skills up to industry standards. In this book, you'll encounter: ? Practical guides on how to manage clients, projects, and build your profile ? Methods to effectively showcase your skills and potential to future employers ? An in-depth guide on how to fast-track your future software engineering career--the right way ? Up-to-date collection and suggestions of printed and online resources The future is for the technically savvy. Add How to Transition Into Software Engineering in 120 Days to your cart TODAY!

A highly anticipated book from a world-class authority who has trained on every continent and taught on many corporate campuses, from GTE to Microsoft First book publication of the two critically acclaimed and widely used testing methodologies developed by the author, known as MITs and S-curves, and more methods and metrics not previously available to the public Presents practical, hands-on testing skills that can be used everyday in real-life development tasks Includes three in-depth case studies that demonstrate how the tests are used Companion Web site includes sample worksheets, support materials, a discussion group for readers, and links to other resources

Fundamentals of Dependable Computing for Software Engineers presents the essential elements of computer system dependability. The book describes a comprehensive dependability-engineering process and explains the roles of software and software engineers in computer system dependability. Readers will learn: Why dependability matters What it means for a system to be dependable How to build a dependable software system How to assess whether a software system is adequately dependable The author focuses on the actions needed to reduce the rate of failure to an acceptable level, covering material essential for engineers developing systems with extreme consequences of failure, such as safety-critical systems, security-critical systems, and critical infrastructure systems. The text explores the systems engineering aspects of dependability and provides a framework for engineers to reason and make decisions about software and its dependability. It also offers a comprehensive approach to achieve software dependability and includes a bibliography of the most relevant literature. Emphasizing the software engineering elements of dependability, this book helps software and computer engineers in fields requiring ultra-high levels of dependability, such as avionics, medical devices, automotive electronics, weapon systems, and advanced information systems, construct software systems that are dependable and within budget and time constraints.

This book constitutes the thoroughly refereed post-conference proceedings of the 6th IPM International Conference on Fundamentals of Software Engineering, FSEN 2015, held in Tehran, Iran, in April 2015. The 21 full papers presented in this volume were carefully reviewed and selected from 64 submissions. The topics of interest in FSEN span over all aspects of formal methods, especially those related to advancing the application of formal methods in software industry and promoting their integration with practical engineering techniques.

This new edition of the book, is restructured to trace the advancements made and landmarks achieved in software engineering. The text 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 organised to cover expanded and revised treatment of all software process activities. 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 who are confirmed adopters of the text • PowerPoint slides available online at www.phindia.com/rajibmall to provide integrated learning to the students NEW TO THE FIFTH EDITION • Several rewritten sections in almost every chapter to increase readability • New topics on latest developments, such as agile development using SCRUM, MC/DC testing, quality models, etc. • A large number of additional multiple choice questions and review questions in all the chapters help students to understand the important concepts TARGET AUDIENCE • BE/B.Tech (CS and IT) • BCA/MCA • M.Sc. (CS) • MBA

Practical Handbook to understand the hidden language of computer hardware and softwareDESCRIPTIONThis book teaches the essentials of software engineering to anyone who wants to become an active and independent software engineer expert. It covers all the software engineering fundamentals without forgetting a few vital advanced topics such as software engineering with artificial intelligence, ontology, and data mining in software engineering.The primary goal of the book is to introduce a limited number of concepts and practices which will achieve the following two objectives:Teach students the skills needed to execute a smallish commercial project.Provide students with the necessary conceptual background for undertaking advanced studies in software engineering through courses or on their own.KEY FEATURESThis book contains real-time executed examples along with case studies.Covers advanced technologies that are intersectional with software engineering.Easy and simple language, crystal clear approach, and straight forward comprehensible presentation.Understand what architecture design involves, and where it fits in the full software development life cycle.Learning and optimizing the critical relationships between analysis and design.Utilizing proven and reusable design primitives and adapting them to specific problems and contexts.WHAT WILL YOU LEARNThis book includes only those concepts that we believe are foundational. As executing a software project requires skills in two dimensions-engineering and project management-this book focuses on crucial tasks in these two dimensions and discuss the concepts and techniques that can be applied to execute these tasks effectively. WHO THIS BOOK IS FORThe book is primarily intended to work as a beginner's guide for Software Engineering in any undergraduate or postgraduate program. It is directed towards students who know the program but have not had formal exposure to software engineering.The book can also be used by teachers and trainers who are in a similar state-they know some programming but want to be introduced to the systematic approach of software engineering.TABLE OF CONTENTS1. Introductory Concepts of Software Engineering2. Modelling Software Development Life Cycle3. Software Requirement Analysis and Specification4. Software Project Management Framework5. Software Project Analysis and Design6. Object-Oriented Analysis and Design7. Designing Interfaces & Dialogues and Database Design8. Coding and Debugging9. Software Testing10. System Implementation and Maintenance11. Reliability12. Software Quality13. CASE and Reuse14. Recent Trends and Development in Software Engineering15. Model Questions with AnswersABOUT THE AUTHORHitesh Mohapatra received a B.E. degree in Information Technology from Gandhi Institute of Engineering and Technology, Gunupur, Biju Patnaik University of Technology, Odisha in 2006, and an MTech. Degree in CSE from Govt. College of Engineering and Technology, Bhubaneswar, Biju Patnaik University of Technology, Odisha in 2009. He is currently a full-time PhD scholar at Veer Surendra Sai University of Technology, Burla, India since 2017 and expected to complete by August 2020. He has contributed 10+ research-level papers (SCI/Scopus), eight international/national conferences (Scopus), and a book on C Programming. He has 12+ years of teaching experience both in industry and academia. His current research interests include wireless sensor network, smart city, smart grid, smart transportation, and smart water.

Amiya Kumar Rath received a B.E. degree in computer from Dr Babasaheb Ambedkar Marathwada University, Aurangabad, in 1990, and an M.B.A. degree in systems management from Shivaji University in 1993. He also received an MTech. Degree in computer science from Utkal University in 2001, and a PhD degree in computer science from Utkal University, in 2005, with a focus on embedded systems. He is currently a Professor with the Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla, India. He has contributed over 80 research-level papers to many national and international journals and conferences, authored seven books published by reputed publishers. His research interests include embedded systems, ad hoc networks, sensor network, power minimization, evolutionary computation, and data mining. Currently, deputed as an adviser to the National Assessment and Accreditation Council (NAAC), Bangalore, India.

Practical Handbook to understand the hidden language of computer hardware and software DESCRIPTION This book teaches the essentials of software engineering to anyone who wants to become an active and independent software engineer expert. It covers all the software engineering fundamentals without forgetting a few vital advanced topics such as software engineering with artificial intelligence, ontology, and data mining in software engineering. The primary goal of the book is to introduce a limited number of concepts and practices which will achieve the following two objectives: Teach students the skills needed to execute a smallish commercial project. Provide students with the necessary conceptual background for undertaking advanced studies in software engineering through courses or on their own. KEY FEATURES - This book contains real-time executed examples along with case studies. - Covers advanced technologies that are intersectional with software engineering. - Easy and simple language, crystal clear approach, and straight forward comprehensible presentation. - Understand what architecture design involves, and where it fits in the full software development life cycle. - Learning and optimizing the critical relationships between analysis and design. - Utilizing proven and reusable design primitives and adapting them to specific problems and contexts. WHAT WILL YOU LEARN This book includes only those concepts that we believe are foundational. As executing a software project requires skills in two dimensions—engineering and project management—this book focuses on crucial tasks in these two dimensions and discuss the concepts and techniques that can be applied to execute these tasks effectively. WHO THIS BOOK IS FOR The book is primarily intended to work as a beginner's guide for Software Engineering in any undergraduate or postgraduate program. It is directed towards students who know the program but have not had formal exposure to software engineering. The book can also be used by teachers and trainers who are in a similar state—they know some programming but want to be introduced to the systematic approach of software engineering. TABLE OF CONTENTS 1. Introductory Concepts of Software Engineering 2. Modelling Software Development Life Cycle 3. Software Requirement Analysis and Specification 4. Software Project Management Framework 5. Software Project Analysis and Design 6. Object-Oriented Analysis and Design 7. Designing Interfaces & Dialogues and Database Design 8. Coding and Debugging 9. Software Testing 10. System Implementation and Maintenance 11.Reliability 12. Software Quality 13. CASE and Reuse 14. Recent Trends and Development in Software Engineering 15. Model Questions with Answers

This book presents the analysis, design, documentation, and quality of software solutions based on the OMG UML v2.5. Notably it covers 14 different modelling constructs including use case diagrams, activity diagrams, business-level class diagrams, corresponding interaction diagrams and state machine diagrams. It presents the use of UML in creating a Model of the Problem Space (MOPS), Model of the Solution Space (MOSS) and Model of the Architectural Space (MOAS). The book touches important areas of contemporary software engineering ranging from how a software engineer needs to invariably work in an Agile development environment through to the techniques to model a Cloud-based solution.

This title presents 30 papers on software engineering by David L. Parnas. Topics covered include: software design, social responsibility, concurrency, synchronization, scheduling and the Strategic Defence Initiative ("Star Wars").

The best way to learn software engineering is by understanding its core and peripheral areas. Foundations of Software Engineering provides in-depth coverage of the areas of software engineering that are essential for becoming proficient in the field. The book devotes a complete chapter to each of the core areas. Several peripheral areas are also explained by assigning a separate chapter to each of them. Rather than using UML or other formal notations, the content in this book is explained in easy-to-understand language. Basic programming knowledge using an object-oriented language is helpful to understand the material in this book. The knowledge gained from this book can be readily used in other relevant courses or in real-world software development environments. This textbook educates students in software engineering principles. It covers almost all facets of software engineering, including requirement engineering, system specifications, system modeling, system architecture, system implementation, and system testing. Emphasizing practical issues, such as feasibility studies, this book explains how to add and develop software requirements to evolve software systems. This book was written after receiving feedback from several professors and software engineers. What resulted is a textbook on software engineering that not only covers the theory of software engineering but also presents real-world insights to aid students in proper implementation. Students learn key concepts through carefully explained and illustrated theories, as well as concrete examples and a complete case study using Java. Source code is also available on the book's website. The examples and case studies increase in complexity as the book progresses to help students build a practical understanding of the required theories and applications.

This book constitutes the thoroughly refereed post-conference proceedings of the 9th International Conference on Fundamentals of Software Engineering, FSEN 2021, held virtually and hosted by IPM in May 2021. The 12 full papers and 4 short papers presented in this volume were carefully reviewed and selected from 38 submissions. The topics of

interest in FSEN span over all aspects of formal methods, especially those related to advancing the application of formal methods in the software industry and promoting their integration with practical engineering techniques. The papers are organized in topical sections on coordination, logic, networks, parallel computation, and testing.

While encouraging the use of modeling techniques for sizing, cost and schedule estimation, reliability, risk assessment, and real-time design, the authors emphasize the need to calibrate models with actual data. Explicit guidance is provided for virtually every task that a software engineer may be assigned, and realistic case studies and examples are used extensively to reinforce the topics presented.

Adopt a diagrammatic approach to creating robust real-time embedded systems Key Features Explore the impact of real-time systems on software design Understand the role of diagramming in the software development process Learn why software performance is a key element in real-time systems Book Description From air traffic control systems to network multimedia systems, real-time systems are everywhere. The correctness of the real-time system depends on the physical instant and the logical results of the computations. This book provides an elaborate introduction to software engineering for real-time systems, including a range of activities and methods required to produce a great real-time system. The book kicks off by describing real-time systems, their applications, and their impact on software design. You will learn the concepts of software and program design, as well as the different types of programming, software errors, and software life cycles, and how a multitasking structure benefits a system design. Moving ahead, you will learn why diagrams and diagramming plays a critical role in the software development process. You will practice documenting code-related work using Unified Modeling Language (UML), and analyze and test source code in both host and target systems to understand why performance is a key design-driver in applications. Next, you will develop a design strategy to overcome critical and fault-tolerant systems, and learn the importance of documentation in system design. By the end of this book, you will have sound knowledge and skills for developing real-time embedded systems. What you will learn Differentiate between correct, reliable, and safe software Discover modern design methodologies for designing a real-time system Use interrupts to implement concurrency in the system Test, integrate, and debug the code Demonstrate test issues for OOP constructs Overcome software faults with hardware-based techniques Who this book is for If you are interested in developing a real-time embedded system, this is the ideal book for you. With a basic understanding of programming, microprocessor systems, and elementary digital logic, you will achieve the maximum with this book. Knowledge of assembly language would be an added advantage.

This book constitutes the thoroughly refereed post-conference proceedings of the 7th International Conference on Fundamentals of Software Engineering, FSEN 2017, held in Tehran, Iran, in April 2017. The 16 full papers presented in this volume were carefully reviewed and selected from 49 submissions. The topics of interest in FSEN span over all aspects of formal methods, especially those related to advancing the application of formal methods in software industry and promoting their integration with practical engineering techniques.

This essential textbook presents a concise introduction to the fundamental principles of software engineering, together with practical guidance on how to apply the theory in a real-world, industrial environment. The wide-ranging coverage encompasses all areas of software design, management, and quality. Topics and features: presents a broad overview of software engineering, including software lifecycles and phases in software development, and project management for software engineering; examines the areas of requirements engineering, software configuration management, software inspections, software testing, software quality assurance, and process quality; covers topics on software metrics and problem solving, software reliability and dependability, and software design and development, including Agile approaches; explains formal methods, a set of mathematical techniques to specify and derive a program from its specification, introducing the Z specification language; discusses software process improvement, describing the CMMI model, and introduces UML, a visual modelling language for software systems; reviews a range of tools to support various activities in software engineering, and offers advice on the selection and management of a software supplier; describes such innovations in the field of software as distributed systems, service-oriented architecture, software as a service, cloud computing, and embedded systems; includes key learning topics, summaries and review questions in each chapter, together with a useful glossary. This practical and easy-to-follow textbook/reference is ideal for computer science students seeking to learn how to build high quality and reliable software on time and on budget. The text also serves as a self-study primer for software engineers, quality professionals, and software managers.

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.

This book discusses important topics for engineering and managing software startups, such as how technical and business aspects are related, which complications may arise and how they can be dealt with. It also addresses the use of scientific, engineering, and managerial approaches to successfully develop software products in startup companies. The book covers a wide

range of software startup phenomena, and includes the knowledge, skills, and capabilities required for startup product development; team capacity and team roles; technical debt; minimal viable products; startup metrics; common pitfalls and patterns observed; as well as lessons learned from startups in Finland, Norway, Brazil, Russia and USA. All results are based on empirical findings, and the claims are backed by evidence and concrete observations, measurements and experiments from qualitative and quantitative research, as is common in empirical software engineering. The book helps entrepreneurs and practitioners to become aware of various phenomena, challenges, and practices that occur in real-world startups, and provides insights based on sound research methodologies presented in a simple and easy-to-read manner. It also allows students in business and engineering programs to learn about the important engineering concepts and technical building blocks of a software startup. It is also suitable for researchers at different levels in areas such as software and systems engineering, or information systems who are studying advanced topics related to software business.

This book provides selective, in-depth coverage of the fundamentals of software engineering by stressing principles and methods through rigorous formal and informal approaches. In contrast to other books which are based on the lifecycle model of software development, the authors emphasize identifying and applying fundamental principles that are applicable throughout the software lifecycle. This emphasis enables readers to respond to the rapid changes in technology that are common today. Principles and techniques are emphasized rather than specific tools--users learn why particular techniques should or should not be used. Understanding the principles and techniques on which tools are based makes mastering a variety of specific tools easier. KEY TOPICS: The authors discuss principles such as design, specification, verification, production, management and tools. Now coverage includes: more detailed analysis and explanation of object-oriented techniques; the use of Unified Modeling Language (UML); requirements analysis and software architecture; Model checking--a technique that provides automatic support to the human activity of software verification; GQM--used to evaluate software quality and help improve the software process; Z specification language. MARKET: For software engineers.

In today's fast and competitive world, a program's performance is just as important to customers as the features it provides. This practical guide teaches developers performance-tuning principles that enable optimization in C++. You'll learn how to make code that already embodies best practices of C++ design run faster and consume fewer resources on any computer--whether it's a watch, phone, workstation, supercomputer, or globe-spanning network of servers. Author Kurt Guntheroth provides several running examples that demonstrate how to apply these principles incrementally to improve existing code so it meets customer requirements for responsiveness and throughput. The advice in this book will prove itself the first time you hear a colleague exclaim, "Wow, that was fast. Who fixed something?"Locate performance hot spots using the profiler and software timersLearn to perform repeatable experiments to measure performance of code changesOptimize use of dynamically allocated variablesImprove performance of hot loops and functionsSpeed up string handling functionsRecognize efficient algorithms and optimization patternsLearn the strengths--and weaknesses--of C++ container classesView searching and sorting through an optimizer's eyeMake efficient use of C++ streaming I/O functionsUse C++ thread-based concurrency features effectively

With multicore processors now in every computer, server, and embedded device, the need for cost-effective, reliable parallel software has never been greater. By explaining key aspects of multicore programming, Fundamentals of Multicore Software Development helps software engineers understand parallel programming and master the multicore challenge. Accessible to newcomers to the field, the book captures the state of the art of multicore programming in computer science. It covers the fundamentals of multicore hardware, parallel design patterns, and parallel programming in C++, .NET, and Java. It also discusses manycore computing on graphics cards and heterogeneous multicore platforms, automatic parallelization, automatic performance tuning, transactional memory, and emerging applications. As computing power increasingly comes from parallelism, software developers must embrace parallel programming. Written by leaders in the field, this book provides an overview of the existing and up-and-coming programming choices for multicores. It addresses issues in systems architecture, operating systems, languages, and compilers.

The testing market is growing at a fast pace and ISTQB certifications are being increasingly requested, with more than 180,000 persons currently certified throughout the world. The ISTQB Foundations level syllabus was updated in 2011, and this book provides detailed course study material including a glossary and sample questions to help adequately prepare for the certification exam. The fundamental aspects of testing are approached, as is testing in the lifecycles from Waterfall to Agile and iterative lifecycles. Static testing, such as reviews and static analysis, and their benefits are examined as well as techniques such as Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transitions and use cases, along with selected white box testing techniques. Test management, test progress monitoring, risk analysis and incident management are covered, as are the methods for successfully introducing tools in an organization. Contents 1. Fundamentals of Testing. 2. Testing Throughout the Software Life Cycle. 3. Static Techniques (FL 3.0). 4. Test Design Techniques (FL 4.0). 5. Test Management (FL 5.0). 6. Tools support for Testing (FL 6.0). 7. Mock Exam. 8. Templates and Models. 9. Answers to the Questions.

This book constitutes the proceedings of the 5th IPM International Conference on Fundamentals of Software Engineering, FSEN 2013, held in Tehran, Iran, in April 2013. The 17 full papers presented in this volume were carefully reviewed and selected from 65 submissions. The topics of interest in FSEN span over all aspects of formal methods, especially those related to advancing the application of formal methods in software industry and promoting their integration with practical engineering techniques.

As the first book about software culture, this book discusses software culture from three perspectives including historical perspective, the classification of software and software applications. This book takes credit from the view of science and technology development. It analyzed scientific innovations and the social areas promoted following the growth of technology. And according to the fact that information helps to build human cultural form, we proposed the concept and researching method of software culture. The aim of writing this book is to strengthen the connection between software and culture, to replenish knowledge system in the subject of software engineering, and to establish a new area of study that is the culture of software.

Salary surveys worldwide regularly place software architect in the top 10 best jobs, yet no real guide exists to help developers become architects. Until now. This book provides the first comprehensive overview of software architecture's many aspects. Aspiring and existing architects alike will examine architectural characteristics, architectural patterns, component

determination, diagramming and presenting architecture, evolutionary architecture, and many other topics. Mark Richards and Neal Ford—hands-on practitioners who have taught software architecture classes professionally for years—focus on architecture principles that apply across all technology stacks. You'll explore software architecture in a modern light, taking into account all the innovations of the past decade. This book examines: Architecture patterns: The technical basis for many architectural decisions Components: Identification, coupling, cohesion, partitioning, and granularity Soft skills: Effective team management, meetings, negotiation, presentations, and more Modernity: Engineering practices and operational approaches that have changed radically in the past few years Architecture as an engineering discipline: Repeatable results, metrics, and concrete valuations that add rigor to software architecture

Requirements engineering is the process of eliciting individual stakeholder requirements and needs and developing them into detailed, agreed requirements documented and specified in such a way that they can serve as the basis for all other system development activities. In this textbook, Klaus Pohl provides a comprehensive and well-structured introduction to the fundamentals, principles, and techniques of requirements engineering. He presents approved techniques for eliciting, negotiating and documenting as well as validating, and managing requirements for software-intensive systems. The various aspects of the process and the techniques are illustrated using numerous examples based on his extensive teaching experience and his work in industrial collaborations. His presentation aims at professionals, students, and lecturers in systems and software engineering or business applications development. Professionals such as project managers, software architects, systems analysts, and software engineers will benefit in their daily work from the didactically well-presented combination of validated procedures and industrial experience. Students and lecturers will appreciate the comprehensive description of sound fundamentals, principles, and techniques, which is completed by a huge commented list of references for further reading. Lecturers will find additional teaching material on the book's website, www.requirements-book.com.

This book constitutes the thoroughly refereed post-conference proceedings of the 8th International Conference on Fundamentals of Software Engineering, FSEN 2019, held in Tehran, Iran, in May 2019. The 14 full papers and 3 short papers presented in this volume were carefully reviewed and selected from 47 submissions. The topics of interest in FSEN span over all aspects of formal methods, especially those related to advancing the application of formal methods in the software industry and promoting their integration with practical engineering techniques. The papers are organized in topical sections on agent based systems, theorem proving, learning, verification, distributed algorithms, and program analysis.

Software Engineering: Architecture-driven Software Development is the first comprehensive guide to the underlying skills embodied in the IEEE's Software Engineering Body of Knowledge (SWEBOK) standard. Standards expert Richard Schmidt explains the traditional software engineering practices recognized for developing projects for government or corporate systems. Software engineering education often lacks standardization, with many institutions focusing on implementation rather than design as it impacts product architecture. Many graduates join the workforce with incomplete skills, leading to software projects that either fail outright or run woefully over budget and behind schedule. Additionally, software engineers need to understand system engineering and architecture—the hardware and peripherals their programs will run on. This issue will only grow in importance as more programs leverage parallel computing, requiring an understanding of the parallel capabilities of processors and hardware. This book gives both software developers and system engineers key insights into how their skillsets support and complement each other. With a focus on these key knowledge areas, Software Engineering offers a set of best practices that can be applied to any industry or domain involved in developing software products. A thorough, integrated compilation on the engineering of software products, addressing the majority of the standard knowledge areas and topics Offers best practices focused on those key skills common to many industries and domains that develop software Learn how software engineering relates to systems engineering for better communication with other engineering professionals within a project environment

Software Engineering Fundamentals is distinctive in its reportage of such subject matters as real-time software design, software metrics, reliability, planning, testing and integration, cost and schedule estimation, human factors, process sizing, quality assurance, technical management, and risk management. If one takes a look regressively back and indulge into more abstract and theoretical facades of some of the programming language, he may find two reasons to get familiar with it. Initially, these factors almost always dictate critical decisions as to what instruments to use and when to implement. People don't intend to engage in using the inaccurate technology for a piece of work, provided they are devoting themselves to create a large software platform. Besides, tools that are different can keep taking considerable time to settle down. If one has to opt for a new device that is radically different from what he is accustomed to, comprehending the basic principles will ensure a smooth transition.

The free book "Fundamentals of Computer Programming with C#" is a comprehensive computer programming tutorial that teaches programming, logical thinking, data structures and algorithms, problem solving and high quality code with lots of examples in C#. It starts with the first steps in programming and software development like variables, data types, conditional statements, loops and arrays and continues with other basic topics like methods, numeral systems, strings and string processing, exceptions, classes and objects. After the basics this fundamental programming book enters into more advanced programming topics like recursion, data structures (lists, trees, hash-tables and graphs), high-quality code, unit testing and refactoring, object-oriented principles (inheritance, abstraction, encapsulation and polymorphism) and their implementation the C# language. It also covers fundamental topics that each good developer should know like algorithm design, complexity of algorithms and problem solving. The book uses C# language and Visual Studio to illustrate the programming concepts and explains some C# / .NET specific technologies like lambda expressions, extension methods and LINQ. The book is written by a team of developers lead by Svetlin Nakov who has 20+ years practical software development experience. It teaches the major programming concepts and way of thinking needed to become a good software engineer and the C# language in the meantime. It is a great start for anyone who wants to become a skillful software engineer. The books does not teach technologies like databases, mobile and web development, but shows the true way to master the basics of programming regardless of the languages, technologies and tools. It is good for beginners and intermediate developers who want to put a solid base for a successful career in the software engineering industry. The book is accompanied by free video lessons, presentation slides and mind maps, as well as hundreds of exercises and live examples. Download the free C# programming book, videos, presentations and other resources from http://introprogramming.info. Title: Fundamentals of Computer Programming with C# (The Bulgarian C# Programming Book) ISBN: 9789544007737 ISBN-13: 978-954-400-773-7 (9789544007737) ISBN-10: 954-400-773-3 (9544007733) Author: Svetlin Nakov & Co. Pages: 1132 Language:

English Published: Sofia, 2013 Publisher: Faber Publishing, Bulgaria Web site: http://www.introprogramming.info License: CC-Attribution-Share-Alike Tags: free, programming, book, computer programming, programming fundamentals, ebook, book programming, C#, CSharp, C# book, tutorial, C# tutorial; programming concepts, programming fundamentals, compiler, Visual Studio, .NET, .NET Framework, data types, variables, expressions, statements, console, conditional statements, control-flow logic, loops, arrays, numeral systems, methods, strings, text processing, StringBuilder, exceptions, exception handling, stack trace, streams, files, text files, linear data structures, list, linked list, stack, queue, tree, balanced tree, graph, depth-first search, DFS, breadth-first search, BFS, dictionaries, hash tables, associative arrays, sets, algorithms, sorting algorithm, searching algorithms, recursion, combinatorial algorithms, algorithm complexity, OOP, object-oriented programming, classes, objects, constructors, fields, properties, static members, abstraction, interfaces, encapsulation, inheritance, virtual methods, polymorphism, cohesion, coupling, enumerations, generics, namespaces, UML, design patterns, extension methods, anonymous types, lambda expressions, LINQ, code quality, high-quality code, high-quality classes, high-quality methods, code formatting, self-documenting code, code refactoring, problem solving, problem solving methodology, 9789544007737, 9544007733

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

Perspectives on Data Science for Software Engineering presents the best practices of seasoned data miners in software engineering. The idea for this book was created during the 2014 conference at Dagstuhl, an invitation-only gathering of leading computer scientists who meet to identify and discuss cutting-edge informatics topics. At the 2014 conference, the concept of how to transfer the knowledge of experts from seasoned software engineers and data scientists to newcomers in the field highlighted many discussions. While there are many books covering data mining and software engineering basics, they present only the fundamentals and lack the perspective that comes from real-world experience. This book offers unique insights into the wisdom of the community's leaders gathered to share hard-won lessons from the trenches. Ideas are presented in digestible chapters designed to be applicable across many domains. Topics included cover data collection, data sharing, data mining, and how to utilize these techniques in successful software projects. Newcomers to software engineering data science will learn the tips and tricks of the trade, while more experienced data scientists will benefit from war stories that show what traps to avoid. Presents the wisdom of community experts, derived from a summit on software analytics Provides contributed chapters that share discrete ideas and technique from the trenches Covers top areas of concern, including mining security and social data, data visualization, and cloud-based data Presented in clear chapters designed to be applicable across many domains

Good software design is simple and easy to understand. Unfortunately, the average computer program today is so complex that no one could possibly comprehend how all the code works. This concise guide helps you understand the fundamentals of good design through scientific laws—principles you can apply to any programming language or project from here to eternity. Whether you're a junior programmer, senior software engineer, or non-technical manager, you'll learn how to create a sound plan for your software project, and make better decisions about the pattern and structure of your system. Discover why good software design has become the missing science Understand the ultimate purpose of software and the goals of good design Determine the value of your design now and in the future Examine real-world examples that demonstrate how a system changes over time Create designs that allow for the most change in the environment with the least change in the software Make easier changes in the future by keeping your code simpler now Gain better knowledge of your software's behavior with more accurate tests

This practically-focused textbook provides a concise and accessible introduction to the field of software testing, explaining the fundamental principles and offering guidance on applying the theory in an industrial environment. Topics and features: presents a brief history of software quality and its influential pioneers, as well as a discussion of the various software lifecycles used in software development; describes the fundamentals of testing in traditional software engineering, and the role that static testing plays in building quality into a product; explains the process of software test planning, test analysis and design, and test management; discusses test outsourcing, and test metrics and problem solving; reviews the tools available to support software testing activities, and the benefits of a software process improvement initiative; examines testing in the Agile world, and the verification of safety critical systems; considers the legal and ethical aspects of software testing, and the importance of software configuration management; provides key learning topics and review questions in every chapter, and supplies a helpful glossary at the end of the book. This easy-to-follow guide is an essential resource for undergraduate students of computer science seeking to learn about software testing, and how to build high quality and reliable software on time and on budget. The work will also be of interest to industrialists including software engineers, software testers, quality professionals and software managers, as well as the motivated general reader.

Software Engineering FundamentalsOxford University Press, USA