Software Verification Examples Computers Engineering

Software Engineering for Science provides an in-depth collection of peer-reviewed chapters that describe experiences with applying software engineering practices to the development of scientific software. It provides a better understanding of how software engineering is and should be practiced, and which software engineering practices are effective for scientific software. The book starts with a detailed overview of the Scientific Software Lifecycle, and a general overview of the scientific software development process. It highlights key issues commonly arising during scientific software development, as well as solutions to these problems. The second part of the book provides examples of the use of testing in scientific software development, including key issues and challenges. The chapters then describe solutions and case studies aimed at applying testing to scientific software development efforts. The final part of the book provides examples of applying software engineering techniques to scientific software, including not only computational modeling, but also software for data management and analysis. The authors describe their experiences and lessons learned from developing complex scientific software in different domains. About the Editors Jeffrey Carver is an Associate Professor in the Department of Computer Science at the University of Alabama. He is one of the primary organizers of the workshop series on Software Engineering for Science (http://www.SE4Science.org/workshops). Neil P. Chue Hong is Director of the Software Sustainability Institute at the University of Edinburgh. His research interests include barriers and incentives in research software ecosystems and the role of software as a research object. George K. Thiruvathukal is Professor of Computer Science at Loyola University Chicago and Visiting Faculty at Argonne National Laboratory. His current research is focused on software metrics in open source mathematical and scientific software.

Pharmaceutical Computer Validation Introduction gives you a comprehensive introduction to computer systems validation as the computers come to life while the head of computer systems at a pharmaceutical company has to prepare for an FDA inspection. You will learn about regulations, the personnel responsible for computer validation, how to accomplish validation, examples of regulatory problems, and so on. It is also relevant for the medical device, food, and cosmetic industries. 86 pages in the guide include a handy printout of several relevant FDA documents. Those readers who wish to have an accompanying program with video and interactivity should also purchase the CD version.

This book presents the thoroughly refereed and revised post-workshop proceedings of the 16th Monterey Workshop, held in Redmond, WA, USA, in March/April 2010. The theme of the workshop was Foundations of Computer Software, with a special focus on Modeling, Development, and Verification of Adaptive Systems. The 13 revised full papers presented were carefully reviewed and selected from numerous submissions for inclusion in the book. The contributions show how the foundations and development techniques of computer software could be adapted even for industrial safety-critical and business-critical applications to improve dependability and robustness and to ensure information privacy and security.

The use of mathematical methods in the development of software is essential when reliable systems are sought; in particular they are now strongly recommended by the official norms adopted in the production of critical software. Program Verification is the area of computer science that studies mathematical methods for checking that a program conforms to its specification. This text is a self-contained introduction to program verification using logic-based methods, presented in the broader context of formal methods for software engineering. The idea of specifying the behaviour of individual software components by attaching contracts to them is now a widely followed approach in program development, which has given rise notably to the development of a number of behavioural interface specification languages and program verification tools. A foundation for the static verification of programs based on contract-annotated routines is laid out in the book. These can be independently verified, which provides a modular approach to the verification of software. The text assumes only basic knowledge of standard mathematical concepts that should be familiar to any computer science student. It includes a self-contained introduction to propositional logic and first-order reasoning with theories, followed by a study of program verification that combines theoretical and practical aspects - from a program logic (a variant of Hoare logic for programs containing user-provided annotations) to the use of a realistic tool for the verification of C programs (annotated using the ACSL specification language), through the generation of verification conditions and the static verification of runtime errors.

From Theory to Practice

Introduction to Software Testing

Modeling, Development, and Verification of Adaptive Systems 16th Monterey Workshop 2010, Redmond, USA, WA, USA, March 31--April 2, Revised Selected Papers

IEEE Software Engineering Standards and Examples

Guide for Implementing a Software Verification and Validation (V&V) Plan

Software Testing and Quality Assurance

In a world permeated by digital technology, engineering is involved in every aspect of human life. Engineers address a wider range of design problems than ever before, raising new questions and challenges regarding their work, as boundaries between engineering, management, politics, education and art disappear in the face of comprehensive socio-technical systems. It is therefore necessary to review our understanding of engineering practice, expertise and responsibility. This book advances the idea that the future of engineering will not be driven by a static view of a closed discipline, but rather will result from a continuous dialogue between different stakeholders involved in the design and application of technical artefacts. Based on papers presented at the 2016 conference of the forum for Philosophy, Engineering and Technology (fPET) in Nuremberg, Germany, the book features contributions by philosophers, engineers and managers from academia and industry, who discuss current and upcoming issues in engineering from a wide variety of different perspectives. They cover topics such as problem solving strategies and value-sensitive design, experimentation and simulation, engineering knowledge and education, interdisciplinary collaboration, sustainability, risk and privacy. The different contributions in combination draw a comprehensive picture of efforts worldwide to come to terms with engineering, its foundations in philosophy, the ethical problems it causes, and its effect on the ongoing development of society.

The purpose of the 13th International Conference on Computer and Information Science (SNPD 2012) held on August 8-10, 2012 in Kyoto, Japan was to bring together researchers and scientists, businessmen and entrepreneurs, teachers and students to discuss the numerous fields of computer science, and to share ideas and information in a meaningful way. Our conference officers selected the best 17 papers from those papers accepted for presentation at the conference in order to publish them in this volume. The papers were chosen based on review scores submitted by members of the program committee, and underwent further rounds of rigorous review. The conference organizers selected 17 outstanding papers from SNPD 2012, all of which you will find in this volume of Springer’s Studies in Computational Intelligence.

"This book explores different applications in V & V that spawn many areas of software development -including real time applications- where V & V techniques are required, providing in all cases examples of the applications"--Provided by publisher.

This book is designed for use as an introductory software engineering course or as a reference for programmers. Up-to-date text uses both theory applications to design reliable, error-free software. Includes a companion CD-ROM with source code third-party software engineering applications.

An Engineering and Scientific Approach

Pharmaceutical Computer Systems Validation

A Software-testing Perspective

An Introduction to Program Verification

Catalog of National Bureau of Standards Publications, 1966-1976

The Essence of Software Engineering

Advances in scientific computing have made modelling and simulation an important part of the decision-making process in engineering, science, and public policy. This book provides a comprehensive and systematic development of the basic concepts, principles, and procedures for verification and validation of models and simulations. The emphasis is placed on models that are described by partial differential and integral equations and the simulations that result from their numerical solution. The methods described can be applied to a wide range of technical fields, from the physical sciences, engineering and technology and industry, through to environmental regulations and safety, product and plant safety, financial investing, and governmental regulations. This book will be genuinely welcomed by researchers, practitioners, and decision makers in a broad range of fields, who seek to improve the credibility and reliability of simulation results. It will also be appropriate either for university courses or for independent study.

Practical Software Engineering presents an introduction to software engineering for a first course. Using the C language, the text stresses the themes of software development by teams; the importance of maintenance; reusability; complete and correct documentation; testing throughout the life cycle; and the use of (CASE) computer-aided software engineering tools to boost productivity. The use of dialogues and a continuous case study enhances understanding of the concepts presented. The text is intended for sophomore to senior level students being introduced to software engineering in computer science, management information systems (MIS), data processing, or wherever students are new to the subject.

Extensively class-tested, this textbook takes an innovative approach to software testing: it defines testing as the process of applying a few well-defined, general-purpose test criteria to a structure or model of the software. It incorporates the latest innovations in testing, including techniques to test modern types of software such as OO, web applications, and embedded software. The book contains numerous examples throughout. An instructor's solution manual, PowerPoint slides, sample syllabi, additional examples and updates, testing tools for students, and example software programs in Java are available on an extensive website.

Comprehensive and up-to-date, it covers the most vital part of software development, independent verification and validation. Presents a variety of methods that will ensure better quality, performance, cost and reliability of technical products and systems. Features numerous hints, tips and instructions for better interaction between verification and validation personnel, development engineers and managers. Includes 8 case histories ranging from major engineering systems through information systems. Many of the principles involved also apply to computer hardware as well as the fields of science and engineering.

NBS Special Publication

Consolidated Reprint of Citations and Abstracts from NBS SP305 and Its Supplements 1-8

Software Quality Engineering

Advances in Computers

Independent Verification and Validation

Quality Assurance, Risk Management and Regulatory Compliance

Thoroughly revised to include the latest industry developments, the Second Edition presents a comprehensive overview of computer validation and verification principles and how to put them into practice. To provide the current best practice and guidance on identifying and implementing improvements for computer systems, the text extensively reviews r

Static analysis of software with deductive methods is a highly dynamic field of research on the verge of becoming a mainstream technology in software engineering. It consists of a large portfolio of - mostly fully automated - analyses: formal verification, test generation, security analysis, visualization, and debugging. All of them are realized in the state-of-art deductive verification framework KeY. This book is the definitive guide to KeY that lets you explore the full potential of deductive software verification in practice. It contains the complete theory behind KeY for active researchers who want to understand it in depth or use it in their own work. But the book also features fully self-contained chapters on the Java Modeling Language and on Using KeY that require nothing else than familiarity with Java. All other chapters are accessible for graduate students (M.Sc. level and beyond). The KeY framework is free and open software, downloadable from the book companion website which contains also all code examples mentioned in this book.

This volume contains the post-proceedings of the 9th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2014, held in Telč, Czech Republic, in October 2014. The 13 thoroughly revised papers were carefully selected out of 28 submissions and are presented together with 4 invited papers. The topics covered by the papers include: algorithms, logic, and games; high performance computing; computer aided analysis, verification, and testing; hardware design and diagnostics; computer graphics and image processing; and artificial intelligence and natural language processing.

This accessible introduction demonstrates a range of testing techniques in the context of a single worked example that runs throughout. Students can easily see the strengths and limitations of progressively more complex approaches in theory and practice. Test automation and the process of testing are emphasised.

An Essential Toolkit for Modern VLSI Design

The Future of Engineering

9th International Doctoral Workshop, MEMICS 2014, Telč, Czech Republic, October 17--19, 2014, Revised Selected Papers

Practical Software Engineering

Verification of Computer Codes in Computational Science and Engineering

Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2012

Professionals in the interdisciplinary field of computer science focus on the design, operation, and maintenance of computational systems and software. Methodologies and tools of engineering are utilized alongside computer applications to develop efficient and precise information databases. Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications is a comprehensive reference source for the latest scholarly material on trends, techniques, and uses of various technology applications and examines the benefits and challenges of these computational developments. Highlighting a range of pertinent topics such as utility computing, computer security, and information systems applications, this multi-volume book is ideally designed for academicians, researchers, students, web designers, software developers, and practitioners interested in computer systems and software engineering.

The one resource needed to create reliable software This text offers a comprehensive and integrated approach tosoftware quality engineering. By following the author's clearguidance, readers learn how to master the techniques to producehigh-quality, reliable software, regardless of the softwaresystem's level of complexity. The first part of the publication introduces major topics insoftware quality engineering and presents quality planning as anintegral part of the process. Providing readers with a solidfoundation in key concepts and practices, the book moves on tooffer in-depth coverage of software testing as a primary means toensure software quality; alternatives for quality assurance,including defect prevention, process improvement, inspection,formal verification, fault tolerance, safety assurance, and damagecontrol; and measurement and analysis to close the feedback loopfor quality assessment and quantifiable improvement. The text's approach and style evolved from the author's hands-onexperience in the classroom. All the pedagogical tools needed tofacilitate quick learning are provided: * Figures and tables that clarify concepts and provide quick topicsummaries * Examples that illustrate how theory is applied in real-worldsituations * Comprehensive bibliography that leads to in-depth discussion ofspecialized topics * Problem sets at the end of each chapter that test readers'knowledge This is a superior textbook for software engineering, computerscience, information systems, and electrical engineering students,and a dependable reference for software and computer professionalsand engineers.

A superior primer on software testing and quality assurance, from integration to execution and automation This important new work fills the pressing need for a user-friendly text that aims to provide software engineers, software quality professionals, software developers, and students with the fundamental developments in testing theory and common testing practices. Software Testing and Quality Assurance: Theory and Practice equips readers with a solid understanding of: Practices that support the production of quality software Software testing techniques Life-cycle models for requirements, defects, test cases, and test results Process models for units, integration, system, and acceptance testing How to build test teams, including recruiting and retaining test engineers Quality Models, Capability Maturity Model, Testing Maturity Model, and Test Process Improvement Model Expertly balancing theory with practice, and complemented with an abundance of pedagogical tools, including test questions, examples, teaching suggestions, and chapter summaries, this book is a valuable, self-contained tool for professionals and an ideal introductory text for courses in software testing, quality assurance, and software engineering.

How can one be assured that computer codes that solve differential equations are correct? Standard practice using benchmark testing no longer provides full coverage because today's production codes solve more complex equations using more powerful algorithms. By verifying the order-of-accuracy of the numerical algorithm implemented in the code, one can detect most any coding mistake that would prevent correct solutions from being computed. Verification of Computer Codes in Computational Science and Engineering sets forth a powerful alternative called OVMSP: Order-Verification via the Manufactured Solution Procedure. This procedure has two primary components: using the Method of Manufactured Exact Solutions to create analytic solutions to the fully-general differential equations solved by the code and using grid convergence studies to confirm the order-of-accuracy. The authors present a step-by-step procedural guide to OVMSP implementation and demonstrate its effectiveness. Properly implemented, OVMSP offers an exciting opportunity to identify virtually all coding 'bugs' that prevent correct solution of the governing partial differential equations. Verification of Computer Codes in Computational Science and Engineering shows you how this can be done. The treatment is clear, concise, and suitable both for developers of production quality simulation software and as a reference for computational science and engineering professionals.

Software Engineer's Reference Book

Formal Verification

Composing Software Components

Essentials of Software Testing

Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications

Software Verification and Validation

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.

Advanced Techniques in Computing Sciences and Software Engineering includes a set of rigorously reviewed world-class manuscripts addressing and detailing state-of-the-art research projects in the areas of Computer Science, Software Engineering, Computer Engineering, and Systems Engineering and Sciences. Advanced Techniques in Computing Sciences and Software Engineering includes selected papers form the conference proceedings of the International Conference on Systems, Computing Sciences and Software Engineering (SCSS 2008) which was part of the International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE 2008).

This book fills the critical need for an in-depth technical reference providing the methods and techniques for building and maintaining confidence in many varities of system software. The intent is to help develop reliable answers to such critical questions as: 1) Are we building the right software for the need? and 2) Are we building the software right? Software Verification and Validation: An Engineering and Scientific Approach is structured for research scientists and practitioners in industry. The book is also suitable as a secondary textbook for advanced-level students in computer science and engineering.

This open access book includes contributions by leading researchers and industry thought leaders on various topics related to the essence of software engineering and their application in industrial projects. It offers a broad overview of research findings dealing with current practical software engineering issues and also pointers to potential future developments. Celebrating the 20th anniversary of

adesso AG, adesso gathered some of the pioneers of software engineering including Manfred Broy, Ivar Jacobson and Carlo Ghezzi at a special symposium, where they presented their thoughts about latest software engineering research and which are part of this book. This way it offers readers a concise overview of the essence of software engineering, providing valuable insights into the latest methodological research findings and adesso's experience applying these results in real-world projects.

Testing, Quality Assurance, and Quantifiable Improvement

Mathematical and Engineering Methods in Computer Science

Validation, Verification, and Testing of Computer Software

Deductive Software Verification – The KeY Book

Theory and Practice

Advanced Techniques in Computing Sciences and Software Engineering

In the last few years we have all become daily users of Internet banking, social networks and cloud services. Preventing malfunctions in these services and protecting the integrity of private data from cyber attack are both current preoccupations of society at large. While modern technologies have dramatically improved the quality of software, the computer science community continues to address the problems of security by developing a theory of formal verification; a body of methodologies, algorithms and software tools for finding and eliminating bugs and security hazards. This book presents lectures delivered at the NATO Advanced Study Institute (ASI) School Marktoberdorf 2015 – 'Verification and Synthesis of Correct and Secure Systems'. During this two-week summer school, held in Marktoberdorf, Germany, in August 2015, the lecturers provided a comprehensive view of the current state-of-the-art in a large variety of subjects, including: models and techniques for analyzing security protocols; parameterized verification; synthesis of reactive systems; software model checking; composition checking; programming by examples; verification of current software; two-player zero-sum games played on graphs; software security by information flow; equivalents – combinatorics; and analysis of synthesis with 'Big Code'. The Marktoberdorf ASIs have become a high-level scientific nucleus of the international scientific network on formal methods, and one of the major international computer science summer schools. This book will be of interest to all those seeking an overview of current theories and applications in formal verification and security.

Basic Approach Foundations of Software Testing is the premiere example-based text and reference for establishing sound engineering practices in test generation, selection, minimization and enhancement, for software projects ranging from the most simple to the highly complex, to those used by government agencies such as the FAA. Foundations of Software Testing also covers data-flow based adequacy and mutation-based adequacy, which are the most powerful of the available test adequacy criteria. It distills knowledge developed by hundreds of testing researchers and practitioners from all over the world and brings it to readers in an easy to understand form. Test generation, selection, priortization and assessment lie at the foundation of all technical activities that arise in a test process. Appropriate deployment of the elements of this strong foundation enables the testing of different types of software applications, including Object Oriented systems, Web services, graphical user interfaces, embedded systems, as well as properties relating to security, performance, and reliability. With over 200 examples and exercises of mathematical, step-by-step approaches, Foundations describes a wide variety of testing techniqes, including finite state models, combinatorial designs, and minimization for regression testing. Table of Contents Part I: PRELIMINARIES 1. Basics of Software Testing Part II: TEST GENERATION 2. Test Generation from Requirements 3. Test Generation from Finite-State Models 4. Test Generation from Combinatorial Designs 5. Test Selection, Minimization and Prioritization for Regression Testing Part III: TEST ADEQUACY ASSESSMENT AND ENHANCEMENT 6. Test-Adequacy: Assessment Using Control Flow and Data Flow 7. Test Adequacy Assessment Using Program Mutation About the Author Aditya P. Mathur is Professor and Head, Department of Computer Science, at Purdue University. He is one of the founders of the department of Computer Science at BITS, Pilani, India where he designed, developed, and taught the first course on microprocessors to undergraduate students from his seminal book Introduction to Microprocessors. Dr. Mathur has been a prolific researcher with over 100 published works in international journals and conferences. His key contributions include a multilingual computer, the saturation effect in software testing, a theory of software cybernetics, and novel techniques for the estimation of software reliability. Students, practitioners, and researchers will find this book an excellent source of simple to advanced techniques to use and improve their knowledge of and expertise in software testing. Praise for Foundations of Software Testing: "The book describes techniques in a lucid manner with great clarity with the help of numerous examples. Illustration of the techniques through appropriate examples makes the book very easy to study and assimilate the deep concepts and thus a unique book in the area of software testing.", Ashish Kundu, Graduate Student, Department of Computer Science, Purdue University. " As a teacher of software testing and validation, I had to search for books that can be used as references in my class and I found that "Foundations of Software Testing" is the best one for at least the following reasons: - It covers a wide range of concepts related to software testing. - It introduces the different concepts smoothly with examples illustrating them. This helps students a lot in understanding the ideas behind each concept introduced. - The exercises at the end of each chapter test if the students understood the concepts properly and as expected. - The references of the book and the discussion at the end of each chapter both give the reader an opportunity to learn more. The slides are well prepared and organized. This facilitates the task of the professor when lecturing.", Professor Abdeslam En-nouaary, Concordia University. "This book teaches software testing as a science and not as an art. It not only presents an engineering approach for handling different testing tasks but, also sets up the formal framework for the presented technique. Thus when compared to other books on testing it can be readily used as a resource by both practitioners and researchers which in my view is the real strength of this book. Initially I thought that there is still much that can be added to this book, but seeing the list of chapters that would be added in subsequent volumes I believe that for the complete set of volumes it would be very difficult to suggest drastic improvements.", Ammar Masood, Graduate student, Department of Electrical and Computer Engineering, Purdue University. "So far, I like your book. Plenty of definitions and terminology that is clearly presented." Christine Ayers, undergraduate student, UT Dallas.

Formal Verification: An Essential Toolkit for Modern VLSI Design presents practical approaches for design and validation, with hands-on advice to help working engineers integrate these techniques into their work. Formal Verification (FV) enables a designer to directly analyze and mathematically explore the quality or other aspects of a Register Transfer Level (RTL) design without using simulations. This can reduce time spent validating designs and more quickly reach a final design for manufacturing. Building on a basic knowledge of SystemVerilog, this book demystifies FV and presents the practical applications that are bringing it into mainstream design and validation processes at Intel and other companies. After reading this book, readers will be prepared to introduce FV in their organization and effectively deploy FV techniques to increase design and validation productivity. Learn formal verification algorithms to gain full coverage without exhaustive simulation Understand formal verification tools and how they differ from simulation tools Create instant test benches to gain insight into how models work and find initial bugs Learn from Intel insiders sharing their hard-won knowledge and solutions to complex design problems

Advances in Computers carries on a tradition of excellence, presenting detailed coverage of innovations in computer hardware, software, theory, design, and applications. The book provides contributors with a medium in which they can explore their subjects in greater depth and breadth than journal articles typically allow. The articles included in this book will become standard references, with lasting value in this rapidly expanding field. Presents detailed coverage of recent innovations in computer hardware, software, theory, design, and applications Includes in-depth surveys and tutorials on new computer technology pertaining to computing: combinatorial testing, constraint-based testing, and black-box testing Written by well-known authors and researchers in the field Includes extensive bibliographies with most chapters Presents volumes devoted to single themes or subfields of computer science

Concepts, Methodologies, Tools, and Applications

Verification, Validation and Testing in Software Engineering

Philosophical Foundations, Ethical Problems and Application Cases

Foundations of Software Testing

Fundamentals of Dependable Computing for Software Engineers

Pharmaceutical Computer Validation Introduction Guidebook

Software Engineer's Reference Book provides the fundamental principles and general approaches, contemporary information, and applications for developing the software of computer systems. The book is comprised of three main parts, an epilogue, and a comprehensive index. The first part covers the theory of computer science and relevant mathematics. Topics under this section include logic, set theory, Turing machines, theory of computation, and computational complexity. Part II is a discussion of software development methods, techniques and technology primarily based around a conventional view of the software life cycle. Topics discussed include methods such as CORE, SSADM, and SREM, and formal methods including VDM and Z. Attention is also given to other technical activities in the life cycle including testing and prototyping. The final part describes the techniques and standards which are relevant in producing particular classes of application. The text will be of great use to software engineers, software project managers, and students of computer science.

A comprehensive treatment of systems and software testing using state of the art methods and tools This book provides valuable insights into state of the art software testing methods and explains, with examples, the statistical and analytic methods used in this field. Numerous examples are used to provide understanding in applying these methods to real-world problems. Leading authorities in applied statistics, computer science, and software engineering present state-of-the-art methods addressing challenges faced by practitioners and researchers involved in system and software testing. Methods include: machine learning, Bayesian methods, graphical models, experimental design, generalized regression, and reliability modeling. Analytic Methods in Systems and Software Testing presents its comprehensive collection of methods in four parts: Part I: Testing Concepts and Methods; Part II: Statistical Models; Part III: Testing Infrastructures; and Part IV: Testing Applications. It seeks to maintain a focus on analytic methods, while at the same time offering a contextual landscape of modern engineering, in order to introduce related statistical and probabilistic models used in this domain. This makes the book an incredibly useful tool, offering interesting insights on challenges in the field for researchers and practitioners alike. Compiles cutting-edge methods and examples of analytical approaches to systems and software testing from leading authorities in applied statistics, computer science, and software engineering Combines methods and examples focused on the analytic aspects of systems and software testing Covers logistic regression, machine learning, Bayesian methods, graphical models, experimental design, generalized regression, and reliability models Written by leading researchers and practitioners in the field, from diverse backgrounds including research, business, government, and consulting Stimulates research at the theoretical and practical level Analytic Methods in Systems and Software Testing is an excellent advanced reference directed toward industrial and academic readers whose work in systems and software development approaches or surpasses existing frontiers of testing and validation procedures. It will also be valuable to post-graduate students in computer science and mathematics.

Software components and component-based software development (CBSD) are acknowledged as the best approach for constructing quality software at reasonable cost. Composing Software Components: A Software-testing Perspective describes a 10-year investigation into the underlying principles of CBSD. By restricting attention to the simplest cases, startling results are obtained: • Components are tested using only executable code. Their behavior is recorded and presented graphically. • Functional and non-functional behavior of systems synthesized from components are calculated from component tests alone. No access to components themselves is required. • Fast, accurate tools support every aspect of CBSD from design through debugging. Case studies of CBSD also illuminate software testing in general, particularly an expanded role for unit testing and the treatment of non-functional software properties. This unique book: • Contains more than a dozen case studies of fully worked-out component synthesis, with revealing insights into fundamental testing issues. • Presents an original, fundamental theory of component composition that includes persistent state and concurrency, based on functional software testing rather than proof-of-programs. • Comes with free supporting software with tutorial examples and data for replication of examples. The Perl software has been tested on Linux, Macintosh, and Windows platforms. Full documentation is provided. • Includes anecdotes and insights from the author's 50-year career in computing as systems programmer, manager, researcher, and teacher. Composing Software Components: A Software-testing Perspective will help software researchers and practitioners to understand the underlying principles of component testing. Advanced students in computer science, engineering, and mathematics can also benefit from the book as a supplemental text and reference.

This is a package of Agent GXP FDA Part 11 and Pharmaceutical Computer Validation Introduction. These two related titles will give the learner an excellent introduction to computer issues in the pharmaceutical industry. Agent GXP FDA Part 11 teaches the FDA regulations on electronic signatures and records in the context of a spoof on a hostage rescue supervised by Pharm Mission Control. The many difficult regulations of Part 11 are broken down into episodes that make the learning more memorable. This thorough section will teach you the history of Part 11, the regulations of Part 11, the implementation of Part 11, the applications of Part 11, the ideas behind Part 11 in order to apply them to new situations, and how to prepare for enforcement of Part 11. This is particularly important for both pharmaceutical/medical device manufacturing and clinical research personnel in FDA-regulated industries, and provides an excellent glimpse of the issues that are likely to face HIPAA implementation of electronic records security measures. This course has been used by thousands of people in the pharmaceutical industry. Pharmaceutical Computer Validation Introduction gives you a comprehensive introduction to computer systems validation as the computers come to life while the head of computer systems at a pharmaceutical company has to prepare for an FDA inspection. You will learn about regulations, the personnel responsible for computer validation, how to accomplish validation, examples of regulatory problems, and so on. It is also relevant for the medical device, food, and cosmetic industries. 224 pages in the manual include handy printouts of many relevant FDA regulations. For convenience, the CD contains the text of some of the regulations. Those readers who wish to have an accompanying program with video and interactivity should also purchase the CD version.

A Life Cycle Engineering Process for Quality Software

Rigorous Software Development

Analytic Methods in Systems and Software Testing

Publications of the National Institute of Standards and Technology ... Catalog

Verification and Validation in Scientific Computing

Software Engineering and Testing