Testing Object Oriented Systems Models Patterns And Tools

KIERA DESHAWN

Object-oriented Construction Handbook
Pearson Education
This book constitutes the thoroughly refereed and peer-reviewed outcome of the Formal Methods and Testing (FORTEST) network - formed as a network established under UK EPSRC funding that investigated the relationships between formal (and semi-formal) methods and software testing - now being a subject group of two BCS Special Interest Groups: Formal Aspects of Computing Science (BCS FACS) and Special Interest Group in Software Testing (BCS SIGIST). Each of the 12 chapters in this book describes a way in which the study of formal methods and software testing can be combined in a manner that brings the benefits of formal methods (e.g., precision, clarity, provability) with the advantages of testing (e.g., scalability, generality, applicability).
27th IFIP WG 6.1 International Conference, ICTSS 2015, Sharjah and Dubai, United Arab Emirates, November 23-25, 2015, Proceedings
Addison-Wesley Professional
MDA Distilled is an accessible introduction to the MDA standard and its tools and technologies. The book describes the fundamental features of MDA, how they fit together, and how you can use them in your organization today. You will also learn how to define a model-driven process for a project involving multiple platforms, implement that process, and then test the resulting system.
Addison-Wesley Professional
Industrial development of software systems needs to be guided by recognized engineering principles. Commercial-off-the-shelf (COTS) components enable the systematic and cost-effective reuse of prefabricated tested parts, a characteristic approach of mature engineering disciplines. This reuse necessitates a thorough test of these components to make sure that each works as specified in a real context. Beydeda and Gruhn invited leading researchers in the area of component testing to contribute to this monograph, which covers all related aspects from testing components in a context-independent manner through testing components in the context of a

specific system to testing complete systems built from different components. The authors take the viewpoints of both component developers and component users, and their contributions encompass functional requirements such as correctness and functionality compliance as well as non-functional requirements like performance and robustness. Overall this monograph offers researchers, graduate students and advanced professionals a unique and comprehensive overview of the state of the art in testing COTS components and COTS-based systems.

MDA Distilled Springer Science & Business Media

This book constitutes the refereed proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering, FASE 2005, held in Edinburgh, UK in April 2005 as part of ETAPS. The 25 revised full papers presented together with an invited paper were carefully reviewed and selected from 105 submissions. The papers are organized in topical sections on Web services, graph grammars and graph transformations, components, product lines, theory, code understanding and validation, UML, and automatic proofs and provers.

Life Cycle Solutions Testing Object-oriented SystemsModels, Patterns, and Tools

Discusses how to define and organize use cases that model the user requirements of a software application. The approach focuses on identifying all the parties who will be using the system, then writing detailed use case descriptions and structuring the use case model. An ATM example runs throughout the book. The authors work at Rational Software. Annotation copyrighted by Book News, Inc., Portland, OR

An Outcome of the FORTEST Network. Revised Selected Papers Addison-Wesley Professional

This book provides an introduction to practical formal modelling techniques in the context of object-oriented system design. It is aimed at both practising software engineers with some prior experience of object-oriented design/programming and at intermediate or advanced students studying object-oriented design or modelling in a short course. The following features make this book particularly attractive to potential instructors: § The relationship with UML and object-oriented programming makes it easy to integrate with the mainstream computing curriculum. Although the book is about formal methods, it does not have to be treated as a specialist topic. § The use of tools and an accessible modelling language improves student motivation. § The industry-based examples and case studies add to the credibility of the approach. § The light touch approach means that the material appeals to students with a wider range of abilities than is the case in a conventional formal methods text. § Support materials as listed above.

Tools and Techniques. IFIP TC6/WG6.1 13th International Conference on Testing of Communicating Systems (TestCom 2000), August 29–September 1, 2000, Ottawa, Canada Addison-Wesley Professional

Component-based software development, CBSD, is no longer just one more new paradigm in software engineering, but is effectively used in development and practice. So far, however, most of the efforts from the software engineering community have concentrated on the functional aspects of CBSD, leaving aside the treatment of the quality issues and extra-functional properties of software components and component-based systems. The 16 revised chapters presented were carefully reviewed and selected for inclusion in the book; together with an introductory survey, they give a coherent and competent survey of the state of the art in the area. The book - the first to focus on quality issues of components and component-based systems - is organized in topical parts on COTS selection, testing and certification, software component quality models, formal models to quality assessment, and CBSD management.

Fundamental Approaches to Software Engineering Springer Science & Business Media

More than ever, mission-critical and business-critical applications depend on object-oriented (OO) software. Testing techniques tailored to the unique challenges of OO technology are necessary to achieve high reliability and quality. "Testing Object-Oriented Systems: Models, Patterns, and Tools" is an authoritative guide to designing and automating test suites for OO applications. This comprehensive book explains why testing must be model-based and provides in-depth coverage of techniques to develop testable models from state machines, combinational logic, and the Unified Modeling Language (UML). It introduces the test design pattern and presents 37 patterns that explain how to design responsibility-based test suites, how to tailor integration and regression testing for OO code, how to test reusable components and frameworks, and how to develop highly effective test suites from use cases. Effective testing must be automated and must leverage object technology. The author describes how to design and code specification-based assertions to offset testability losses due to inheritance and polymorphism. Fifteen micro-patterns present oracle strategies--practical solutions for one of the hardest problems in test design. Seventeen design patterns explain how to automate your test suites with a coherent OO test harness framework. The author provides thorough coverage of testing issues such as: The bug hazards of OO programming and differences from testing procedural code How to design responsibility-based tests for classes, clusters, and subsystems using class invariants, interface data flow models, hierarchic state machines, class associations, and scenario analysis How to support reuse by effective testing of abstract classes, generic classes, components, and frameworks How to choose an integration strategy that supports iterative and incremental development How to achieve comprehensive system testing with testable use cases How to choose a regression test approach How to develop expected test results and evaluate the post-test state of an object How to automate testing with assertions, OO test drivers, stubs, and test frameworks Real-world experience, world-class best practices, and the latest research in object-oriented testing are included. Practical examples illustrate test design and test automation for Ada 95, C++, Eiffel, Java, Objective-C, and Smalltalk. The UML is used throughout, but the test design patterns apply to systems developed with any OO language or methodology. 0201809389B04062001

Testing of Communicating Systems Cambridge University Press

Intended for both undergraduate and postgraduate students of computer science and engineering, information technology, students of computer applications, and working IT professionals, this text describes the practices necessary for the development of quality software. The contents of the book have been framed based on the syllabi prescribed by different Universities and also covers the topics required for working in the IT industry. Based on the experience of the author in the industry, academics, consultancy and corporate trainings in India and abroad, the book covers the methodologies, techniques, and underlying concepts used in Software Quality Assurance and Testing. The treatment of the topics is crisp and

accompanied with illustrative examples with minimum jargons. Topics of relevance in the industry, which a student must be familiar with before start of a career, are covered in the book. The book also discusses the concepts that a working IT professional should know. The book provides an insight into the tools available for different types of testing. Each chapter contains Quizzes, Multiple Choice Questions and Review Questions which help the readers to qualify in the international certification examinations. Key features • Covers topics relevant to the industry • Concepts discussed in an easy to understand way and illustrated with practical examples and figures wherever required • Contains “Objective Questions” at the end of the book • Includes topics prescribed in international certification exams in Software Quality and Testing

Springer

David A. Sykes is a member of Wofford College's faculty.

Getting Your Models Ready for MDA CRC Press

This volume contains the papers presented at the 12th SDL Forum, Grimstad, Norway. The SDL Forum was ?rst held in 1982, and then every two years from 1985. Initially the Forum was concerned only with the Speci?cation and Description Language that was ?rst standardized in the 1976 Orange Book of the Inter- tional Telecommunication Union (ITU). Since then, many developments took place and the language has undergone several changes. However, the main underlying paradigm has survived, and it is the reason for the success of the Speci?cation and Description Language in many projects. This paradigm is based on the following important principles of distributed - plications: Communication: large systems tend to be described using smaller parts that communicate with each other; State: the systems are described on the basis of an explicit notion of state; State change: the behavior of the system is described in terms of (local) changes of the state. The original language is not the only representative for this kind of paradigm, so the scope of the SDL Forum was extended quite soon after the ?rst few events to also include other ITU standardized languages of the same family, such as MSC, ASN.1 and TTCN. This led to the current scope of System Design Languages coveringallstagesofthedevelopmentproces sincludinginparticularSDL,MSC, UML, ASN.1, eODL, TTCN, and URN. The focus is clearly on the advantages to users, and how to get from these languages the same advantage given by the ITU Speci?cation and Description Language: code generation from high-level speci?cations.

Testing of Communicating Systems Springer Science & Business Media

Testing of Communicating Systems presents the latest international results in both the theory and industrial practice of the testing of communicating systems. The topics discussed range from tools and techniques for testing to test standards, frameworks, notations, algorithms, fundamentals of testing, and industrial experiences and issues. The tools and techniques discussed apply to conformance testing, interoperability testing, performance testing of communications software, Internet protocols and applications, and multimedia and distributed systems in general, such as systems for electronic commerce. This volume contains the extensively refereed proceedings of the 13th International Conference on Testing of Communicating Systems (TestCom 2000), which was sponsored by the International Federation for Information Processing (IFIP) and held in Ottawa, Ontario, Canada in early September 2000. Testing of Communicating Systems is essential reading for engineers, designers, managers of IT products and services, and all researchers interested in advancing the technology of engineering Internet frameworks, systems, services, and applications for reliability and quality.

SDL 2005: Model Driven Springer Science & Business Media

This revised and enlarged edition of a classic in Old Testament scholarship reflects the most up-to-date research on the prophetic books and offers substantially expanded discussions of important new insight on Isaiah and the other prophets.

Testing Software and Systems Springer Science & Business Media

The first UML book to cover Rational Rose 2000, this brand-new edition reviews the three key interrelated components of state-of-the-art software system design: the Rational Unified process, the Unified Modeling Language, and Rational Rose 2000. Then, through a simplified case study, it walks developers through a real-world business system. Includes screen shots demonstrating UML at work in the Rational Rose 2000 modeling tool.

Use Case Modeling Springer

"Designing a large software system is an extremely complicated undertaking that requires juggling differing perspectives and differing goals, and evaluating differing options. Applied Software Architecture is the best book yet that gives guidance as to how to sort out and organize the conflicting pressures and produce a successful design." -- Len Bass, author of Software Architecture in Practice. Quality software architecture design has always been important, but in today's fast-paced, rapidly changing, and complex development environment, it is essential. A solid, well-thought-out design helps to manage complexity, to resolve trade-offs among conflicting requirements, and, in general, to bring quality software to market in a more timely fashion. Applied Software Architecture provides practical guidelines and techniques for producing quality software designs. It gives an overview of software architecture basics and a detailed guide to architecture design tasks, focusing on four fundamental views of architecture--conceptual, module, execution, and code. Through four real-life case studies, this book reveals the insights and best practices of the most skilled software architects in designing software architecture. These case studies, written with the masters who created them, demonstrate how the book's concepts and techniques are embodied in state-of-the-art architecture design. You will learn how to: create designs flexible enough to incorporate tomorrow's technology; use architecture as the basis for meeting performance, modifiability, reliability, and safety requirements; determine priorities among conflicting requirements and arrive at a successful solution; and use software architecture to help integrate system components. Anyone involved in software architecture will find this book a valuable compendium of best practices and an insightful look at the critical role of architecture in software development. 0201325713B07092001

A Foundation for Model-driven Architecture Addison-Wesley Professional

This comprehensive and well-written book presents the fundamentals of object-oriented software engineering and discusses the recent technological developments in the field. It focuses on object-oriented software engineering in the context of an overall effort to present object-oriented concepts, techniques and models that can be applied in software estimation, analysis, design, testing and quality improvement. It applies unified modelling language notations to a series of examples with a real-life case study. The example-oriented approach followed in this book will help the readers in understanding and applying the concepts of object-oriented software engineering quickly and easily in various application

domains. This book is designed for the undergraduate and postgraduate students of computer science and engineering, computer applications, and information technology. KEY FEATURES : Provides the foundation and important concepts of object-oriented paradigm. Presents traditional and object-oriented software development life cycle models with a special focus on Rational Unified Process model. Addresses important issues of improving software quality and measuring various object-oriented constructs using object-oriented metrics. Presents numerous diagrams to illustrate object-oriented software engineering models and concepts. Includes a large number of solved examples, chapter-end review questions and multiple choice questions along with their answers.

Validated Designs for Object-oriented Systems PHI Learning Pvt. Ltd.

Teaches readers how to test and analyze software to achieve an acceptable level of quality at an acceptable cost Readers will be able to minimize software failures, increase quality, and effectively manage costs Covers techniques that are suitable for near-term application, with sufficient technical background to indicate how and when to apply them Provides balanced coverage of software testing & analysis approaches By incorporating modern topics and strategies, this book will be the standard software-testing textbook

Testing Commercial-off-the-Shelf Components and Systems Springer

Object-oriented programming (OOP) has been the leading paradigm for developing software applications for at least 20 years. Many different methodologies, approaches, and techniques have been created for OOP, such as UML, Unified Process, design patterns, and eXtreme Programming. Yet, the actual process of building good software, particularly large, interactive, and long-lived software, is still emerging. Software engineers familiar with the current crop of methodologies are left wondering, how does all of this fit together for designing and building software in real projects? This handbook from one of the world's leading software architects and his team of software engineers presents guidelines on how to develop high-quality software in an application-oriented way. It answers questions such as: * How do we analyze an application domain utilizing the knowledge and experience of the users? * What is the proper software architecture for large, distributed interactive systems that can utilize UML and design patterns? * Where and how should we utilize the techniques and methods of the Unified Process and eXtreme Programming? This book brings together the best of research, development, and day-to-day project work. "The strength of the book is that it focuses on the transition from design to implementation in addition to its overall vision about software development." -Bent Bruun Kristensen, University of Southern Denmark, Odense

Real-time Design Patterns Addison-Wesley Professional

Software testing is an essential phase in software development, which is the primary way to evaluate software under development. With rapidly growing user needs and the complex design of software application, software testing needs more efficient and effective ways to assure the reliability and quality of software. The supporting technology for software testing has been widely studied, and Unified Modeling Language (UML) is one of the technologies which can be powerfully applied in software testing. UML is a practical standard for design and visualization of complex software systems. It is not only helpful for the software designers and developers but also for the software testers. Object-oriented programming and Web Services are the most popular technologies of software development for Object-Oriented systems and web application. However, there are several testing issues unique to Object-Oriented software and Web Services. The characteristics of Object-Oriented language increase the complexity of relationships in software components and introduce new kinds of faults raising issues in software testing. In Service-Oriented Architecture (SOA), the enterprises take advantage of the dynamic discovery and invocation capabilities of Web Services to build loosely coupled Service-Oriented applications. The complex applications can be obtained by discovering and composing existing services, but it also arises many testing issues by the simplistic approach of Web Services. In this dissertation, a framework of UML-based software testing design is proposed to model the Object-Oriented software system and Service-Oriented software for more effective and efficient software testing. The framework consists of three main components; test model generation, test case generation, and testing execution. First, the test model generation uses UML diagrams to create test models for Object-Oriented software systems and Service-Oriented software separately. Second, a test case generation approach that includes defined coverage criteria and generation of the test path and test data according to the test model is introduced. For the test path generation, we proposed an algorithm to automatically generate the paths according to different coverage criteria. Third, the mutation testing and different mutant operators are used for testing execution to verify the proposed test model.

Process, Principles and Techniques Springer

Software testing can be regarded as an art, a craft, and a science. The practical, step-by-step approach presented in this book provides a bridge between these different viewpoints. A single worked example runs throughout, with consistent use of test automation. Each testing technique is introduced in the context of this example, helping students see its strengths and weaknesses. The technique is then explained in more detail, providing a deeper understanding of underlying principles. Finally the limitations of each technique are demonstrated by inserting faults, giving learners concrete examples of when each technique succeeds or fails in finding faults. Coverage includes black-box testing, white-box testing, random testing, unit testing, object-oriented testing, and application testing. The authors also emphasise the process of applying the techniques, covering the steps of analysis, test design, test implementation, and interpretation of results. The book's web site has programming exercises and Java source code for all examples.