Testing Object Oriented Systems Models 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
Overviews the process of building and compiling executable UML models for software development. The book focuses on the BridgePoint tool suite and object action language developed by Project Technology. The authors discuss identifying system requirements, diagramming classes and attributes, constraints on the class diagram, ways of building sets of communicating statechart diagrams, and model verification. Annotation copyrighted by Book News, Inc., Portland, OR.
Object-Oriented Analysis and Design for Information Systems clearly explains real object-oriented programming in practice. Expert author Raul Sidnei Wazlawick explains concepts such as object responsibility, visibility and the real need for delegation in detail. The object-oriented code generated by using these concepts in a systematic way is concise, organized and reusable. The patterns and solutions presented in this book are based in research and industrial applications. You will come away with clarity regarding processes and use cases and a clear understand of how to expand a use case. Wazlawick clearly explains clearly how to build meaningful sequence diagrams. Object-Oriented Analysis and Design for Information Systems illustrates how and why building a class model is not just placing classes into a diagram. You will learn the necessary organizational patterns so that your software architecture will be maintainable. Learn how to build better class models, which are more maintainable and understandable. Write use cases in a more efficient and standardized way, using more effective and less complex diagrams. Build true object-oriented code with division of responsibility and delegation.
Provides complete coverage of the Ada language and Ada programming in general by recognized authorities in Ada software engineering. Demonstrates the power and performance of Ada in the management of large-scale object-oriented systems, and shows how to use Ada features such as generics, packages, and tasking.
14th International Conference, MODELS 2011, Wellington, New Zealand, October 16-21, 2011, Proceedings
Modeling with UML, OCL, and IFML
Life Cycle Solutions
ECOOP 2001 - Object-Oriented Programming
An Outcome of the FORTEST Network. Revised Selected Papers
Robust Scalable Architecture for Real-time Systems
Fundamentals of Object-oriented Design in UML
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
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.
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.
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.

12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009, Proceedings

Testing Commercial-off-the-Shelf Components and Systems

Model-Driven Testing

A Practical Guide to Testing Object-oriented Software

Introduction to Software Testing

Agile Modeling with UML

33nd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings

Written by the original members of an industry standardization group, this book shows you how to use UML to test complex software systems. It is the definitive reference for the only UML-based test specification language, written by the creators of that language. It is supported by an Internet site that provides information on the latest tools and uses of the profile. The authors introduce UTP step-by-step, using a case study that illustrates how UTP can be used for test modeling and test specification.

"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

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.

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.

Formal Methods and Software Engineering
Model Driven Engineering Languages and Systems
Software Engineering with Ada
UML-based Software Testing Design for Object-oriented and Web Service Software System
OBJECT-ORIENTED SOFTWARE ENGINEERING
Applied Software Architecture
SDL 2005: Model Driven

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.

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).

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

bull; Learn to better leverage the siginificant power of UML 2.0 and the Model-Driven Architecture standard bull; The OCL helps developers produce better software by adding vital definition to their designs bull; Updated to reflect the latest version of the standard - OCL 2.0

Process, Principles and Techniques

12th International SDL Forum, Grimstad, Norway, June 20-23, 2005, Proceedings
Applying Use Case Driven Object Modeling with UML
Formal Methods and Testing
Developing Application-oriented Software with the Tools & Materials Approach
Object-Oriented Analysis and Design for Information Systems

This book constitutes the refereed proceedings of the 17th IFIP TC 6/WG 6.1 International Conference on Testing Communicating Systems, TestCom 2005, held in Montreal, Canada in 2005. The 24 revised full papers presented together with the extended abstract of a keynote talk were carefully reviewed and selected from initially 62 submissions. The papers address a

issues in testing communicating systems, ranging from classical telecommunication issues to general software testing.
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 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 testing & analysis approaches By incorporating modern topics and strategies, this book will be the standard software-testing textbook
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 and techniques for testing to test standards, frameworks, notations, algorithms, fundamentals of testing, and industrial experiences and issues. The tools and techniques discussed apply conformance testing, interoperability testing, performance testing of communications software, Internet protocols and applications, and multimedia and distributed systems in general, suc systems for electronic commerce. This volume contains the extensively refereed proceedings of the 13th International Conference on Testing of Communicating Systems (TestCom 2000 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 ess reading for engineers, designers, managers of IT products and services, and all researchers interested in advancing the technology of engineering Internet frameworks, systems, services applications for reliability and quality.
Formal engineering methods are changing the way that software systems are - veloped.Withlanguageandtoolsupport,theyarebeingusedforautomaticcode generation, and for the automati abstraction and checking of implementations. In the future, they will be used at every stage of development: requirements, speci?cation, design, implementation, testing, and documentat ICFEM series of conferences aims to bring together those interested in the application of formal engineering methods to computer systems. Researchers and practitioners, from industry, academia, and government, are encouraged to attend,andtohelpadvancethestateoftheart.Authorsarestronglyencouraged to make their ideas as accessible as possible, and there is a clea upon work that promises to bring practical, tangible bene?t: reports of case studies should have a conceptual message, theory papers should have a clear link to application, and papers describing tools should have an account of results. ICFEM 2004 was the sixth conference in the series, and the ?rst to be held in North America. Previous conferences were held in Singa China, UK, A- tralia, and Japan. The Programme Committee received 110 papers and selected 30forpresentation.The?nalversionsofthosepapersareincludedhere,together with 2-page ab the 5 accepted tutorials, and shorter abstracts for the 4 invited talks.
An Anotated E-commerce Example
Fundamental Approaches to Software Engineering
Doing Hard Time
SOFSEM 2007: Theory and Practice of Computer Science
Getting Your Models Ready for MDA
8th International Conference, FASE 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceeding
15th European Conference, Budapest, Hungary, June 18-22, 2001, Proceedings
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.
"This is the fourth report on mothers and babies in NSW to combine the annual reports of the NSW Midwives Data Collection (MDC), the Neonatal Intensive Care Units' Data Collection and the NSW Birth Defects Register."--Page 9.
The book describes a method for developing the testing of components in parallel with their functionality based on models. UML models are used to derive the testing architecture for an application, the testing interfaces and the component testers. The method provides a process and guidelines for modeling and developing these artifacts. The book also discusses the implications of built-in contract testing with other component-based development technologies such as product-line engineering, middleware platforms, reuse principles etc. Still further, it describes a new method for specifying and checking real-time properties of object-oriented, component-based real-time systems that are based on dynamic execution time analysis with optimization algorithms.
This book focuses on the methodological treatment of UML/P and addresses three core topics of model-based software development: code generation, the systematic testing of programs using a model-based definition of test cases, and the evolutionary refactoring and transformation of models. For each of these topics, it first details the foundational concepts and techniques, and then presents their application with UML/P. This separation between basic principles and applications makes the content more accessible and allows the reader to transfer this knowledge directly to other model-based approaches and languages. After an introduction to the book and its primary goals in Chapter 1, Chapter 2 outlines an agile UML-based approach using UML/P as the primary development language for creating executable models, generating code from the models, designing test cases, and planning iterative evolution through refactoring. In the interest of completeness, Chapter 3 provides a brief summary of UML/P, which is used throughout the book. Next, Chapters 4 and 5 discuss core techniques for code generation, addressing the architecture of a code generator and methods for controlling it, as well as the suitability of UML/P notations for test or product code. Chapters 6 and 7 then discuss general concepts for testing software as well as the special features which arise due to the use of UML/P. Chapter 8 details test patterns to show how to use UML/P diagrams to define test cases and emphasizes in particular the use of functional tests for distributed and concurrent software systems. In closing, Chapters 9 and 10

examine techniques for transforming models and code and thus provide a solid foundation for refactoring as a type of transformation that preserves semantics. Overall, this book will be of great benefit for practical software development, for academic training in the field of Software Engineering, and for research in the area of model-based software development. Practitioners will learn how to use modern model-based techniques to improve the production of code and thus significantly increase quality. Students will find both important scientific basics as well as direct applications of the techniques presented. And last but not least, the book will offer scientists a comprehensive overview of the current state of development in the three core topics it covers.

Developing Real-time Systems with UML, Objects, Frameworks, and Patterns

Software Testing and Analysis

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

Object-oriented Construction Handbook

Model-Based Testing for Embedded Systems

A Foundation for Model-driven Architecture

Validated Designs for Object-oriented Systems

Testing Object-oriented SystemsModels, Patterns, and ToolsAddison-Wesley Professional

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 coveringallstagesofthedevelopmentprocessincludinginparticularSDL,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.

This book constitutes the refereed proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems, MODELS 2011, held in Wellington, New Zealand, in October 2011. The papers address a wide range of topics in research (foundations track) and practice (applications track). For the first time a new category of research papers, vision papers, are included presenting "outside the box" thinking. The foundations track received 167 full paper submissions, of which 34 were selected for presentation. Out of these, 3 papers were vision papers. The application track received 27 submissions, of which 13 papers were selected for presentation. The papers are organized in topical sections on model transformation, model complexity, aspect oriented modeling, analysis and comprehension of models, domain specific modeling, models for embedded systems, model synchronization, model based resource management, analysis of class diagrams, verification and validation, refactoring models, modeling visions, logics and modeling, development methods, and model integration and collaboration.

Doing Hard Time is written to facilitate the daunting process of developing real-time systems. It presents an embedded systems programming methodology that has been proven successful in practice. The process outlined in this book allows application developers to apply practical techniques - garnered from the mainstream areas of object-oriented software development - to meet the demanding qualifications of real-time programming. Bruce Douglass offers ideas that are up-to-date with the latest concepts and trends in programming. By using the industry standard Unified Modeling Language (UML), as well as the best practices from object technology, he guides you through the intricacies and specifics of real-time systems development. Important topics such as schedulability, behavioral patterns, and real-time frameworks are demystified, empowering you to become a more effective real-time programmer.

The Object Constraint Language

Component-Based Software Testing with UML

Principles of Model-driven Architecture
Executable UML
SOFTWARE QUALITY ASSURANCE, TESTING AND METRICS
17th IFIP TC 6/WG 6.1 International Conference, TestCom 2005, Montreal, Canada, May 31 - June 2, 2005, Proceedings
6th International Conference on Formal Engineering Methods, ICFEM 2004, Seattle, WA, USA, November 8-12, 2004, Proceedings

Object-oriented programming increases software reusability, extensibility, interoperability, and reliability. Software testing is necessary to realize these benefits. Software testing aims to uncover as many programming errors as possible at a minimum cost. A major challenge to the software engineering community remains how to reduce the cost and improve the quality of software testing. The requirements for testing object-oriented programs differ from those for testing conventional programs. Testing Object-Oriented Software illustrates these differences and discusses object-oriented software testing problems, focusing on the difficulties and challenges testers face. The book provides a general framework for class- and system-level testing and examines object-oriented design criteria and high testability metrics. It offers object-oriented testing techniques, ideas and methods for unit testing, and object-oriented program integration-testing strategy. Readers are shown how they can drastically reduce regression test costs, presented with steps for object-oriented testing, and introduced to object-oriented test tools and systems. In addition to software testing problems, the text covers various test methods developers can use during the design phase to generate programs with good testability. The book's intended audience includes object-oriented program testers, program developers, software project managers, and researchers working with object-oriented testing.

This book constitutes the refereed proceedings of the 15th European Conference on Object-Oriented Programming, ECOOP 2001, held in Budapest, Hungary, in June 2001. The 18 revised full papers presented together with one invited paper were carefully reviewed and selected from 108 submissions. The book is organized in topical sections on sharing and encapsulation, type inference and static analysis, language design, implementation techniques, reflection and concurrency, and testing and design.

Practical Model-Based Testing gives a practical introduction to model-based testing, showing how to write models for testing purposes and how to use model-based testing tools to generate test suites. It is aimed at testers and software developers who wish to use model-based testing, rather than at tool-developers or academics. The book focuses on the mainstream practice of functional black-box testing and covers different styles of models, especially transition-based models (UML state machines) and pre/post models (UML/OCL specifications and B notation). The steps of applying model-based testing are demonstrated on examples and case studies from a variety of software domains, including embedded software and information systems. From this book you will learn: The basic principles and terminology of model-based testing How model-based testing differs from other testing processes How model-based testing fits into typical software lifecycles such as agile methods and the Unified Process The benefits and limitations of model-based testing, its cost effectiveness and how it can reduce time-to-market A step-by-step process for applying model-based testing How to write good models for model-based testing How to use a variety of test selection criteria to control the tests that are generated from your models How model-based testing can connect to existing automated test execution platforms such as Mercury Test Director, Java JUnit, and proprietary test execution environments Presents the basic principles and terminology of model-based testing Shows how model-based testing fits into the software lifecycle, its cost-effectiveness, and how it can reduce time to market Offers guidance on how to use different kinds of modeling techniques, useful test generation strategies, how to apply model-based testing techniques to real applications using case studies

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

Practical Model-Based Testing
Testing Object-oriented Systems
Use Case Modeling
A Tools Approach
Real-time Design Patterns
Testing of Communicating Systems
Testing Object-Oriented Software

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

This book constitutes the refereed proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2007, held in Harrachov, Czech Republic in January 2007. The 69 revised full papers, presented together with 11 invited contributions were carefully reviewed and selected from 283 submissions. The papers were organized in four topical tracks.

Addressing various aspects of object-oriented software techniques with respect to their impact on testing, this text argues that the testing of object-oriented software is not restricted to a single phase of software development. The book concentrates heavily on the testing of classes and of components or sub-systems, and a major part is devoted to this subject. C++ is used throughout this book that is intended for software practitioners, managers, researchers, students, or anyone interested in object-oriented technology and its impacts throughout the software engineering life-cycle.

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.

Model-Driven Development with Executable UML

Code Generation, Testing, Refactoring

Models, Patterns and Tools

MDA Distilled

Models, Patterns, and Tools

Using the UML Testing Profile

What the experts have to say about Model-Based Testing for Embedded Systems: "This book is exactly what is needed at the exact right time in this fast-growing area. From its beginnings over 10 years ago of deriving tests from UML statecharts, model-based testing has matured into a topic with both breadth and depth. Testing embedded systems is a natural application of MBT, and this book hits the nail exactly on the head. Numerous topics are presented clearly, thoroughly, and concisely in this cutting-edge book. The authors are world-class leading experts in this area and teach us well-used and validated techniques, along with new ideas for solving hard problems. "It is rare that a book can take recent research advances and present them in a form ready for practical use, but this book accomplishes that and more. I am anxious to recommend this in my consulting and to teach a new class to my students." —Dr. Jeff Offutt, professor of software engineering, George Mason University, Fairfax, Virginia, USA "This handbook is the best resource I am aware of on the automated testing of embedded systems. It is thorough, comprehensive, and authoritative. It covers all important technical and scientific aspects but also provides highly interesting insights into the state of practice of model-based testing for embedded systems." —Dr. Lionel C. Briand, IEEE Fellow, Simula Research Laboratory, Lysaker, Norway, and professor at the University of Oslo, Norway "As model-based testing is entering the mainstream, such a comprehensive and intelligible book is a must-read for anyone looking for more information about improved testing methods for embedded systems. Illustrated with numerous aspects of these techniques from many contributors, it gives a clear picture of what the state of the art is today." —Dr. Bruno Legeard, CTO of Smartesting, professor of Software Engineering at the University of Franche-Comt é , Besan ç on, France, and co-author of Practical Model-Based Testing

Fundamentals of Object-Oriented Design in UML shows aspiring and experienced programmers alike how to apply design concepts, the UML, and the best practices in OO development to improve both their code and their success rates with object-based projects.

The pioneering organizers of the ?rst UML workshop in Mulhouse, France inthe summerof1998couldhardlyhaveanticipatedthat,in littleoveradecade, theirinitiativewouldblossomintotoday ’ shighlysuccessfulMODELSconference series, the premier annual gathering of researchersand practitioners focusing on a very important new technical discipline: model-based software and system engineering. This expansion is, of course, a direct consequence of the growing signi?cance and success of model-based methods in practice. The conferences have contributed greatly to the heightened interest in the ?eld, attracting much young talent and leading to the gradualemergence of its correspondingscienti?c and engineering foundations. The proceedings from the MODELS conferences are one of the primary references for anyone interested in a more substantive study of the domain. The 12th conference took place in Denver in the USA, October 4 – 9, 2009 along with numerous satellite workshops and tutorials, as well as several other related scienti?c gatherings. The conference was exceptionally fortunate to have three eminent, invited keynote speakers from industry: Stephen Mellor, Larry Constantine, and Grady Booch.