Formal Methods Specification And Verification Guidebook

B is one of the few formal methods which has robust, commercially-available tool support for the entire development lifecycle from specification through to code generation. This volume provides a comprehensive introduction to the B Abstract Machine Notation, and to how it can be used to support formal specification and development of high integrity systems. A strong emphasis is placed on the use of B in the context of existing software development methods, including object-oriented analysis and design. The text includes a large number of worked examples, graduated exercises in B AMN specification and development (all of which have been class-tested), two extended case studies of the development process, and an appendix of proof techniques suitable for B. Based on material which has been used to teach B at postgraduate and undergraduate level, this volume will provide invaluable reading a wide range of people, including students, project technical managers and workers, and researchers with an interest in methods integration and B semantics.

This book constitutes the thoroughly refereed conference proceedings of the 18th International Workshop on Formal Methods for Industrial Critical Systems, FMICS 2013, held in Madrid, Spain, in September 2013. The 13 papers presented were carefully selected from 25 submissions and cover topics such as design, specification, code generation and testing based on formal methods, methods, techniques and tools to support automated analysis, certification, debugging, learning, optimization and transformation of complex, distributed, dependable, real-time systems and embedded systems, verification and validation methods, tools for the development of formal design descriptions, case studies and experience reports on industrial applications of formal methods, impact of the adoption of formal methods on the development process and associated costs, application of formal methods in standardization and industrial forums.

Formal methods is the term used to describe the specification and verification of software and software systems using mathematical logic. Various methodologies have been developed and incorporated into software tools. An important subclass is distributed systems. There are many books that look at particular methodologies for such systems, e.g. CSP, process algebra. This book offers a more balanced introduction for graduate students that describes the various approaches, their strengths and weaknesses, and when they are best used. Milner's CCS and its operational semantics are introduced, together with notions of behavioural equivalence based on bisimulation techniques and with variants of Hennessy-Milner modal logics. Later in the book, the presented theories are extended to take timing issues into account. The book has arisen from various courses taught in Iceland and Denmark and is designed to give students a broad introduction to the area, with exercises throughout.

Increasingly numerous and complex communication protocols are being employed in distributed systems and computer networks of all types. This Note describes some of the more formal techniques that are being developed to facilitate design of correct protocols. Our major conclusion is that it is vital to specify the services provided by a protocol layer in addition to specifying the cooperating protocol entities which make up the layer. We develop service specifications of several representative protocols by

using formal techniques from software engineering such as abstract machines and buffer histories. A survey of protocol verification methods and a bibliography indexed by key phrases are also provided. (Author).

Subjects addressed include: model-based testing, schemes and patterns of assumption/promise-based system specification, requirements models for critical systems, engineering evolving and self-adaptive systems, unifying models of data flow, model-based verification and analysis of real-time systems, and model checking -- Back cover.

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.

This volume contains the proceedings of FORTE 2008, 28th IFIP WG6.1 - ternational Conference on Formal Techniques for Networked and Distributed Systems. FORTE 2008 was held at the Campus Innovation Center in Tokyo, Japan during June 10–13, 2008. FORTE denotes a series of international wo- ing conferences on formal description techniques applied to computer networks and distributed systems. The conference series started in 1981 under the name PSTV. In 1988 a second series under the name FORTE was set up. Both - ries were united to FORTE/PSTV in 1996. In 2001 the conference changed the name to its current form. Recent conferences of this long series were held in Berlin (2003), Madrid(2004), Taipei(2005), Paris(2006), and Tallinn(2007). As in the previous year, FORTE 2008 was collocated with TESTCOM/ FATES 2008: the 20th IFIP International Conference on Testing of Com- nicating Systems (TESTCOM) and the 8th International Workshop on Formal Approaches to Testing of Software (FATES). The co-location of FORTE and TESTCOM/FATES fostered the collaboration between their communities. The commonspiritofboth conferenceswasunderpinnedby jointopening andclosing sessions, invited talks, as well as joint social events.

This textbook gives students a comprehensive introduction to formal methods and their application in software and hardware

specification and verification. It has three parts: The first part introduces some fundamentals in formal methods, including set theory, functions, finite state machines, and regular expressions. The second part focuses on logi

The Formal Methods Specification and Verification Guidebook for Software and Computer Systems describes a set of techniques called Formal Methods (FM), and outlines their use in the specification and verification of computer systems and software. Development of increasingly complex systems has created a need for improved specification and verification techniques. NASA's Safety and Mission Quality Office has supported the investigation of techniques such as FM, which are now an accepted method for enhancing the quality of aerospace applications. The guidebook provides information for managers and practitioners who are interested in integrating FM into an existing systems development process. Information includes technical and administrative considerations that must be addressed when establishing the use of FM on a specific project. The guidebook is intended to aid decision makers in the successful application of FM to the development of high-quality systems at reasonable cost. This is the first volume of a planned two-volume set. The current volume focuses on administrative and planning considerations for the successful application of FM. Unspecified Center...

This book constitutes the refereed proceedings of the 13th International Conference on Formal Engineering Methods, ICFEM 2011, held in Durham, UK, October 2011. The 40 revised full papers together with 3 invited talks presented were carefully reviewed and selected from 103 submissions. The papers address all current issues in formal methods and their applications in software engineering. They are organized in topical sections on formal models; model checking and probability; specification and development; security; formal verification; cyber physical systems; event-B; verification, analysis and testing; refinement; as well as theorem proving and rewriting.

An approach to software design that introduces a fully automated analysis giving designers immediate feedback, now featuring the latest version of the Alloy language. In Software Abstractions Daniel Jackson introduces an approach to software design that draws on traditional formal methods but exploits automated tools to find flaws as early as possible. This approach—which Jackson calls “lightweight formal methods” or “agile modeling”—takes from formal specification the idea of a precise and expressive notation based on a tiny core of simple and robust concepts but replaces conventional analysis based on theorem proving with a fully automated analysis that gives designers immediate feedback. Jackson has developed Alloy, a language that captures the essence of software abstractions simply and succinctly, using a minimal toolkit of mathematical notions. This revised edition updates the text, examples, and appendixes to be fully compatible with Alloy 4.

This book constitutes the refereed proceedings of the TFM 2009, held in Eindhoven, The Netherlands in November 2009. The 10 revised full papers presented together with an abstracts of invited talk were carefully reviewed and selected from 19 submissions. The papers presented explore the experiences of teaching FMs, both successful and unsuccessful, educational resources including the use of books, case studies and the internet, the education of weak and mathphobic students, the integration, or otherwise, of FMs into the curriculum, including, contributions to the definition of a Formal Methods Body of Knowledge (FMBOK),

the advantages of FM-trained graduates in the workplace, changing attitudes towards FMs in students, academic staff and practitioners and the necessary mathematical background.

Design and Analysis of Distributed Embedded Systems is organized similar to the conference. Chapters 1 and 2 deal with specification methods and their analysis while Chapter 6 concentrates on timing and performance analysis. Chapter 3 describes approaches to system verification at different levels of abstraction. Chapter 4 deals with fault tolerance and detection. Middleware and software reuse aspects are treated in Chapter 5. Chapters 7 and 8 concentrate on the distribution related topics such as partitioning, scheduling and communication. The book closes with a chapter on design methods and frameworks.

NATO's Division of Scientific and Environmental Affairs sponsored this Advan ced Study Institute because it was felt to be timely to cover this important and challengjng subject for the first time in the framework of NATO's ASI programme. The significance of real-time systems in everyones' life is rapidly growing. The vast spectrum of these systems can be characterised by just a few examples of increasing complexity: controllers in washing machines, air traffic control systems, control and safety systems of nuclear power plants and, finally, future military systems like the Strategic Defense Initiative (SDI). The import ance of such systems for the well-being of people requires considerable efforts in research and development of highly reliable real-time systems. Furthermore, the competitiveness and prosperity of entire nations now depend on the early app lication and efficient utilisation of computer integrated manufacturing systems (CIM), of which real-time systems are an essential and decisive part. Owing to its key significance in computerised defence systems, real-time computing has also a special importance for the Alliance. The early research and development activities in this field in the 1960s and 1970s aimed towards improving the then unsatisfactory software situation. Thus, the first high-level real-time languages were defined and developed: RTL/2, Coral 66, Procol, LTR, and PEARL. In close connection with these language develop ments and with the utilisation of special purpose process control peripherals, the research on real-time operating systems advanced considerably.

This is a graduate-level introduction to formal methods. The first part presents two formal languages: logic, in various forms, and Communicating Sequential Process (CSP) as a process algebra. The second part offers specification and testing methods for formal development of software. Building on the foundations from the first part, the reader is allowed to embrace methods for practical applications. The reader will find the examples cutting across chapters valuable for this purpose. The final section takes the reader further into application domains.

In computing science design plays an eminently important role. By now, it is quite clear that the issue of proper design of programs within a formal calculus is one of the most interesting and most difficult parts of computing science. Many demanding problems have to be envisaged here such as notations, rules and calculi, and the study of semantic models. We are 'far away from comprehensive and widely accepted solutions in these areas. Discussions at the summer school have clearly shown that people have quite different perspectives and priorities with respect to these three main areas. There is a general agreement that notation is very important. Here, notation is not so much used in the sense of "syntactic sugar", but rather in the sense of abstract syntax, in

the sense of language constructs. Proper notation can significantly improve our understanding of the nature of the objects that we are dealing with and simplify the formal manipulation of these objects. However, influenced by educational background, habits, and schools of thought there are quite different tastes with respect to notation. The papers in these proceedings show very clearly how different those notations can be even when talking about quite similar objects.

Circuits and architectures have become more complex in terms of structure, interconnection topology, and data flow. Design correctness has become increasingly significant, as errors in design may result in strenuous debugging, or even in the repetition of a costly manufacturing process. Although circuit simulation has been used traditionally and widely as the technique for checking hardware and architectural designs, it does not guarantee the conformity of designs to specifications. Formal methods therefore become vital in guaranteeing the correctness of designs and have thus received a significant amount of attention in the CAD industry today.This book presents a formal method for specifying and verifying the correctness of systolic array designs. Such architectures are commonly found in the form of accelerators for digital signal, image, and video processing. These arrays can be quite complicated in topology and data flow. In the book, a formalism called STA is defined for these kinds of dynamic environments, with a survey of related techniques. A framework for specification and verification is established. Formal verification techniques to check the correctness of the systolic networks with respect to the algorithmic level specifications are explained. The book also presents a Prolog-based formal design verifier (named VSTA), developed to automate the verification process, as using a general purpose theorem prover is usually extremely time-consuming. Several application examples are included in the book to illustrate how formal techniques and the verifier can be used to automate proofs.

This book constitutes the refereed proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2004, held in Austin, Texas, USA in November 2004. The 29 revised full papers presented together with the abstract of an invited talk were carefully reviewed and selected from 69 submissions. The papers address all current issues on tools, methods, algorithms, and foundational theory for the application of formalized reasoning to all aspects of computer-aided systems design, including specification, verification, synthesis, and testing.

This book presents the revised versions of nine invited lectures presented by leading researchers at the fourth edition of the International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFT 2004, held in Bertinoro, Italy, September 2004. SFM 2004 is devoted to real-time systems. The lectures presented cover formal models and languages for the specification, modeling, analysis, and verification of time-critical systems, the expressiveness of such models and languages, as well as supporting tools and related applications in different domains. The book offers a unique and comprehensive state-of-the-art survey on real-time systems. Researchers and advanced students will appreciate the book as a valuable source of reference and a systematic guide to the use of formal methods for the specification, analysis, and verification of real-time systems.

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).

Based on material used by the authors in their teaching, this volume provides a detailed comparison and study of the various methods for reasoning about software. The material offers a comprehensive understanding of which program structures are easier to manipulate by formal techniques, thus allowing professionals to write programs that are easier to reason about informally. The basic technology presented should be of use in all programming environments.

This extensively revised and updated new edition of Specification of Software Systems builds upon the original focus on software specification with added emphasis on the practice of formal methods for specification and verification activities for different types of software systems and at different stages of developing software systems. Topics and features: provides a wide coverage of formal specification techniques and a clear writing style, supported by end-of-chapter bibliographic notes for further reading; presents a logical structure, with sections devoted to specification fundamentals, basics of formalism, logic, set theory and relations, property-oriented specification methods, and model-based specification techniques; contains end-of-chapter exercises and numerous case studies, with potential course outlines suggested in the Preface; covers Object-Z, B-Method, and Calculus of Communicating Systems; offers material that can be taught with tool-supported laboratory projects.

This volume is the outcome of deliberations on formal methods in aerospace. The book specially delves into the use of formal methods for verification, validation, and optimization of software in safety critical and time critical applications, such as those in aerospace engineering. The chapters in this book are authored by leading corporate and government R&D scientists. The contents of this book will be useful to researchers and professionals alike.

Growing demands for the quality, safety, and security of software can only be satisfied by the rigorous application of formal methods during software design. This book methodically investigates the potential of first-order logic automated theorem provers for applications in software engineering. Illustrated by complete case studies on protocol verification, verification of security protocols, and logic-based software reuse, this book provides techniques for assessing the prover's capabilities and for selecting and developing an appropriate interface architecture.

This is the final report prepared under SBIR Phase II Contract N00014-95-C-0131 from the Office of Naval Research. It reports on the development of a set of software tools for specification and verification of distributed real time systems using formal methods. The task of this SBIR Phase II effort was to create a commercial-strength CASE toolset suitable for handling real-life verification problems. A preceding Phase I contract was concerned with development of the approach to the problem and design of a prototype environment 14. Forthcoming Phase III work will demonstrate the utility of the toolset to potential customers and establish it as a commercial off- the-shelf (COTS) specification and verification product. It is important to note that commercialization work, which is the task of Phase III efforts, has already begun during the current Phase II period (see Section IV). The toolset has been given the name PARAGON, which stands for 'Process-Algebraic Real-time Analysis with Graphics-Oriented Notation'. The toolset is indented to be used by designers of real-time systems for early detection of errors. The mathematical complexity of formal specification and verification has been hidden from the end users as much as possible. To achieve this, the specification language uses notions used by designers in their work as primitives. This provides for concise specifications, readable even by a non-specialist.

Formal methods have been applied successfully to the verification of medium-sized programs in protocol and hardware design. However, their application to more complex systems, resulting from the object-oriented and the more recent component-based software engineering paradigms, requires further development of specification and verification techniques supporting the concepts of reusability and modifiability. This book presents revised tutorial lectures given by invited speakers at the Second International Symposium on Formal Methods for Components and Objects, FMCO 2003, held in Leiden, The Netherlands, in November 2003. The 17 revised lectures by leading researchers present a comprehensive account of the potential of formal methods applied to large and complex software systems such as component-based systems and object systems. The book makes a unique contribution to bridging the gap between theory and practice in software engineering.

This is an excellent introduction to formal methods which will bring anyone who needs to know about this important topic up to speed. It is comprehensive, giving the reader all the information needed to explore the field of formal methods in more detail. It offers: a guide to the mathematics required; comprehensive but easy-to-understand introductions to various methods; a run-down of how formal methods can help to develop high-quality systems that come in on time, within budget, and according to requirements.

This book constitutes the refereed proceedings of the 10th International Conference on Formal Engineering Methods, ICFEM 2008, held in Kitakyushu-City, Japan, October 2008. The 20 revised full papers together with 3 invited talks presented were carefully reviewed and selected from 62 submissions. The papers address all current issues in formal methods and their applications in software engineering. They are organized in topical sections on specification and verification; testing; verification; model checking and analysis; tools; application of formal methods; semantics.

This book is a solid foundation of the most important formalisms used for specification and verification of reactive systems. In particular, the text presents all important results on m-calculus, w-automata, and temporal logics, shows the relationships between these formalisms and describes state-of-the-art verification procedures for them. It also discusses advantages and disadvantages of these formalisms, and shows up their strengths and weaknesses. Most results are given with detailed proofs, so that the presentation is almost self-contained. Includes all definitions without relying on other material Proves all theorems in detail Presents detailed algorithms in pseudo-code for verification as well as translations to other formalisms

Industrial Strength Formal Methods in Practice provides hands-on experience and guidance for anyone who needs to apply formal methods successfully in an industrial context. Each chapter is written by an expert in software engineering or formal methods, and contains background information, introductions to the techniques being used, actual fragments of formalised components, details of results and an analysis of the overall approach. It provides specific details on how to produce high-quality software that comes in on-time and within budget. Aimed mainly at practitioners in software engineering and formal methods, this book will also be of interest to the following groups; academic researchers working in formal methods who are interested in evidence of their success and in how they can be applied on an industrial scale, and students on advanced software engineering courses who need real-life

specifications and examples on which to base their work.

This book presents 8 papers accompanying the lectures of leading researchers given at the 6th edition of the International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2006). SFM 2006 was devoted to formal techniques for hardware verification and covers several aspects of the hardware design process, including hardware design languages and simulation, property specification formalisms, automatic test pattern generation, symbolic trajectory evaluation, and more.

This book constitutes the thoroughly refereed post-workshop proceedings of the 7th International Workshop on Web Services and Formal Methods, WS-FM 2010, held in Hoboken, NJ, USA, in September 2010. The 11 revised full papers presented together with one invited paper were carefully reviewed and selected from 26 submissions. The papers feature topics such as web services; service oriented computing; cloud computing; formal methods; verification specification; testing; and business process management.

Real-time systems need to react to certain input stimuli within given time bounds. For example, an airbag in a car has to unfold within 300 milliseconds in a crash. There are many embedded safety-critical applications and each requires real-time specification techniques. This text introduces three of these techniques, based on logic and automata: duration calculus, timed automata, and PLC-automata. The techniques are brought together to form a seamless design flow, from real-time requirements specified in the duration calculus; via designs specified by PLC-automata; and into source code for hardware platforms of embedded systems. The syntax, semantics, and proof methods of the specification techniques are introduced; their most important properties are established; and real-life examples illustrate their use. Detailed case studies and exercises conclude each chapter. Ideal for students of real-time systems or embedded systems, this text will also be of great interest to researchers and professionals in transportation and automation.

Hybrid systems are models for complex physical systems and have become a widely used concept for understanding their behavior. Many applications are safety-critical, including car, railway, and air traffic control, robotics, physical–chemical process control, and biomedical devices. Hybrid systems analysis studies how we can build computerized controllers for physical systems which are guaranteed to meet their design goals. The author gives a unique, logic-based perspective on hybrid systems analysis. It is the first book that leverages the power of logic for hybrid systems. The author develops a coherent logical approach for systematic hybrid systems analysis, covering its theory, practice, and applications. It is further shown how the developed verification techniques can be used to study air traffic and railway control systems. This book is intended for researchers, postgraduates, and professionals who are interested in hybrid systems analysis, cyberphysical or embedded systems design, logic and theorem proving, or transportation and

automation.

Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume 1; Planning and Technology InsertionCreatespace Independent Publishing Platform

This volume comprises the papers selected for presentation at the international conference on Formal Methods in Programming and Their Applications, held in Academgorodok, Novosibirsk, Russia, June-July 1993. The conference was organized by the Institute of Informatics Systems of the Siberian Division of the Russian Academy of Sciences and was the first forum organized by the Institute which was entirely dedicated to formal methods. The main scientific tracks of the conference were centered around formal methods of program development and program construction. The papers in the book are grouped into the following parts: - formal semantics methods - algebraic specification methods - semantic program analysis and abstract interpretation - semantics of parallelism - logic of programs - software specification and verification - transformational development and program synthesis.

The two-volume set LNCS 7609 and 7610 constitutes the thoroughly refereed proceedings of the 5th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, held in Heraklion, Crete, Greece, in October 2012. The two volumes contain papers presented in the topical sections on adaptable and evolving software for eternal systems, approaches for mastering change, runtime verification: the application perspective, model-based testing and model inference, learning techniques for software verification and validation, LearnLib tutorial: from finite automata to register interface programs, RERS grey-box challenge 2012, Linux driver verification, bioscientific data processing and modeling, process and data integration in the networked healthcare, timing constraints: theory meets practice, formal methods for the developent and certification of X-by-wire control systems, quantitative modelling and analysis, software aspects of robotic systems, process-oriented geoinformation systems and applications, handling heterogeneity in formal development of HW and SW Systems.