Formal Methods In Software Engineering Examples

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

This book constitutes the refereed proceedings of the 10th International Conference on Software Engineering and Formal Methods, SEFM 2012, held in Thessaloniki, Greece, in October 2012. The 19 revised research papers presented together with 3 short papers, 2 tool papers, and 2 invited talks were carefully reviewed and selected from 98 full submissions. The SEFM conference aspires to advance the state-of-the-art in formal methods, to enhance their scalability and usability with regards to their application in the software industry and to promote their integration with practical engineering methods.

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.

This book constitutes the refereed proceedings of the 18th International Conference on Formal Engineering Methods, ICFEM 2016, held in Tokyo, Japan, in November 2016. The 27 revised full papers presented together with three invited

talks were carefully reviewed and selected from 64
submissions. The conference focuses in all areas related to
formal engineering meth-ods, such as verification and
validation, software engineering, formal specification and
modeling, software security, and software reliability.
This is a gently-paced software engineering text that
concentrates on the use of formal methods for the
development of high integrity software. It assumes no
previous knowledge of formal methods, and presents a
practical and comprehensive treatment of the subject suitable
for second and third year undergraduates on computing
courses, and students on taught master's courses. The book
is packed with examples and exercises throughout.
By presenting state-of-the-art research results on various
aspects of formal and visual modeling of software and
systems, this book commemorates the 60th birthday of
Hartmut Ehrig. The 24 invited reviewed papers are written by
students and collaborators of Hartmut Ehrig who are
established researchers in their fields. Reflecting the scientific
interest and work of Hartmut Ehrig, the papers fall into three
main parts on graph transformation, algebraic specification
and logic, and formal and visual modeling.
Based around a theme of the construction of a game engine,
this textbook is for final year undergraduate and graduate
students, emphasising formal methods in writing robust code
quickly. This book takes an unusual, engineering-inspired
approach to illuminate the creation and verification of large
software systems . Where other textbooks discuss business
practices through generic project management techniques or
detailed rigid logic systems, this book examines the
interaction between code in a physical machine and the logic
applied in creating the software. These elements create an
informal and rigorous study of logic, algebra, and geometry
through software. Assuming prior experience with C, C++, or

Java programming languages, chapters introduce UML, OCL, and Z from scratch. Extensive worked examples motivate readers to learn the languages through the technical side of software science.

The art, craft, discipline, logic, practice and science of developing large-scale software products needs a professional base. The textbooks in this three-volume set combine informal, engineeringly sound approaches with the rigor of formal, mathematics-based approaches. This volume covers the basic principles and techniques of specifying systems and languages. It deals with modelling the semiotics (pragmatics, semantics and syntax of systems and languages), modelling spatial and simple temporal phenomena, and such specialized topics as modularity (incl. UML class diagrams), Petri nets, live sequence charts, statecharts, and temporal logics, including the duration calculus. Finally, the book presents techniques for interpreter and compiler development of functional, imperative, modular and parallel programming languages. This book is targeted at late undergraduate to early graduate university students, and researchers of programming methodologies. Vol. 1 of this series is a prerequisite text.

This book constitutes the proceedings of the 22nd International Conference on Formal Engineering Methods, ICFEM 2020, held in Singapore, Singapore, in March 2021. The 16 full and 4 short papers presented together with 1 doctoral symposium paper in this volume were carefully reviewed and selected from 41 submissions. The papers cover theory and applications in formal engineering methods together with case studies. They also represent the recent development in the

use and development of formal engineering methods for software and system development.

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.

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.
Formal Methods for Software
EngineeringLanguages, Methods, Application
DomainsSpringer
This book constitutes the refereed proceedings of
the 14th International Conference on Formal
Engineering Methods, ICFEM 2012, held in Kyoto,
Japan, November 2012. The 31 revised full papers
together with 3 invited talks presented were carefully
reviewed and selected from 85 submissions. The
papers address all current issues in formal methods
and their applications in software engineering. They

are organized in topical sections on concurrency, applications of formal methods to new areas, quantity and probability, formal verification, modeling and development methodology, temporal logics, abstraction and refinement, tools, as well as testing and runtime verification.

This book constitutes the refereed proceedings of the 16th International Conference on Formal Engineering Methods, ICFEM 2014, held in Luxembourg, Luxembourg, in November 2014. The 28 revised full papers presented were carefully reviewed and selected from 73 submissions. The papers cover a wide range of topics in the area of formal methods and software engineering and are devoted to advancing the state of the art of applying formal methods in practice. They focus in particular on combinations of conceptual and methodological aspects with their formal foundation and tool support. Static analysis of software with deductive methods is a highly dynamic field of research on the verge of becoming a mainstream technology in software engineering. It consists of a large portfolio of - mostly fully automated - analyses: formal verification, test generation, security analysis, visualization, and debugging. All of them are realized in the state-of-art deductive verification framework KeY. This book is the definitive guide to KeY that lets you explore the full potential of deductive software verification in practice. It contains the complete theory behind KeY

for active researchers who want to understand it in depth or use it in their own work. But the book also features fully self-contained chapters on the Java Modeling Language and on Using KeY that require nothing else than familiarity with Java. All other chapters are accessible for graduate students (M.Sc. level and beyond). The KeY framework is free and open software, downloadable from the book companion website which contains also all code examples mentioned in this book.
Formal Methods Fact File VDM and Z Andrew Harry Formal methods provide a means of specifying computer systems that is unambiguous,concise and well suited to the development of complex software systems for which accuracy and reliability are critical. Heavily mathematical and seemingly difficult to learn, for many they hold little appeal. Andrew Harry speaks as a programmer who has travelled the difficult route to an understanding of formal methods techniques, and knows why it's worth the effort. He explains, in refreshingly simple terms, what formal methods are, why we need them, what should motivate our choice of methods and how to use them effectively. The book presents a novel view of formal methods, spanning the range of specification techniques. An overview of the different styles of formal notation is followed by detailed chapters on the two most popular languages, VDM and Z, consistent with the latest draft standards. There is a

readable account of the underlying maths, a short introduction to semantics for proof, and a survey of tools available. Teaching aids include quick reference appendices on the notation and syntax of VDM and Z; exercises (and their solutions); and a useful glossary of terms. A more populist account than most, this book's "informal" treatment of the subject will appeal to students and industrial programmers who want to know more but find little on the shelves for the novice. Visit our Web page! http://www.wiley.com/compbooks/
This invaluable textbook/reference provides an easy-to-read guide to the fundamentals of formal methods, highlighting the rich applications of formal methods across a diverse range of areas of computing. Topics and features: introduces the key concepts in software engineering, software reliability and dependability, formal methods, and discrete mathematics; presents a short history of logic, from Aristotle's syllogistic logic and the logic of the Stoics, through Boole's symbolic logic, to Frege's work on predicate logic; covers propositional and predicate logic, as well as more advanced topics such as fuzzy logic, temporal logic, intuitionistic logic, undefined values, and the applications of logic to AI; examines the Z specification language, the Vienna Development Method (VDM) and Irish School of VDM, and the unified modelling language (UML); discusses Dijkstra's calculus of weakest

preconditions, Hoare's axiomatic semantics of programming languages, and the classical approach of Parnas and his tabular expressions; provides coverage of automata theory, probability and statistics, model checking, and the nature of proof and theorem proving; reviews a selection of tools available to support the formal methodist, and considers the transfer of formal methods to industry; includes review questions and highlights key topics in every chapter, and supplies a helpful glossary at the end of the book. This stimulating guide provides a broad and accessible overview of formal methods for students of computer science and mathematics curious as to how formal methods are applied to the field of computing.

Software Engineering with OBJ: Algebraic Specification in Action is a comprehensive introduction to OBJ, the most widely used algebraic specification system. As a formal specification language, OBJ makes specifications and designs more precise and easier to read, as well as making maintenance easier and more accurate. OBJ differs from most other specification languages not just in having a formal semantics, but in being executable, either through symbolic execution with term rewriting, or more generally through theorem proving. One problem with specifications is that they are often wrong. OBJ can help validate specifications by executing test cases, and by

proving properties. As well as providing a detailed introduction to the language and the OBJ system that implements it, Software Engineering with OBJ: Algebraic Specification in Action provides case studies by leading practitioners in the field, in areas such as computer graphics standards, hardware design, and parallel computation. The case studies demonstrate that OBJ can be used in a wide variety of ways to achieve a wide variety of practical aims in the system development process. The papers on various OBJ systems also demonstrate that the language is relatively easy to understand, implement, and use, and that it supports formal reasoning in a straightforward but powerful way. Software Engineering with OBJ: Algebraic Specification in Action will be of interest to students and teachers in the areas of data types, programming languages, semantics, theorem proving, and algebra, as well as to researchers and practitioners in software engineering.

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 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).

In any serious engineering discipline, it would be unthinkable to construct a large system without having a precise notion of what is to be built and without verifying how the system is expected to function. Software engineering is no different in this respect. Formal methods involve the use of mathematical notation and calculus in software development; such methods are difficult to apply to large-scale systems with practical constraints (e.g., limited developer skills, time and budget restrictions, changing requirements). Here Liu claims that formal engineering methods may bridge this gap. He advocates the incorporation of mathematical notation into the software engineering process, thus substantially improving the rigor, comprehensibility and effectiveness of the methods commonly used in

industry. This book provides an introduction to the
SOFL (Structured Object-Oriented Formal
Language) method that was designed and industry-
tested by the author. Written in a style suitable for
lecture courses or for use by professionals, there are
numerous exercises and a significant real-world
case study, so the readers are provided with all the
knowledge and examples needed to successfully
apply the method in their own projects.
This book constitutes the refereed proceedings of
the 20th International Conference on Formal
Engineering Methods, ICFEM 2018, held in Gold
Coast, QLD, Australia, in November 2018. The 22
revised full papers presented together with 14 short
papers were carefully reviewed and selected from 66
submissions. The conference focuses on all areas
related to formal engineering methods, such as
verification; network systems; type theory; theorem
proving; logic and semantics; refinement and
transition systems; and emerging applications of
formal methods.
Formal engineering methods are intended to o?er
e?ective means for integ- tion of formal methods and
practical software development technologies in the
context of software engineering. Their purpose is to
provide e?ective, rigorous, and systematic
techniques for signi?cant improvement of software
productivity, quality, and tool supportability. In
comparison with formal methods, a distinct feature of

formal engineering methods is that they emphasize the importance of the balance between the qualities of simplicity, visualization, and preciseness for practicality. To achieve this goal, formal engineering methods must be - veloped on the basis of both formal methods and existing software technologies in software engineering, and they must serve the improvement of the softwa- engineering process. ICFEM 2008 marks the tenth anniversary of the ?rst ICFEM conference, which was held in Hiroshima in 1997. It aims to bring together researchers and practitioners who are interested in the development and application of formal engineering methods to present their latest work and discuss future research directions. The conference o?ers a great opportunity for researchers in both formal methods and software engineering to exchange their ideas, experience, expectation and to ?nd out whether and how their research results can help advance the state of the art.
Formal engineering methods are changing the way that software systems are - veloped.Withlanguagean dtoolsupport,theyarebeingusedforautomaticcode generation, and for the automatic abstraction and checking of implementations. In the future, they will be used at every stage of development: requirements, speci?cation, design, implementation, testing, and documentation. The 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,andtohelpadvancethestateofth eart.Authorsarestronglyencouraged to make their ideas as accessible as possible, and there is a clear emphasis 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 Singapore, China, UK, A- tralia, and Japan. The Programme Committee received 110 papers and selected 30forpresentation.The?nalversionsofth osepapersareincludedhere,together with 2-page abstracts for the 5 accepted tutorials, and shorter abstracts for the 4 invited talks.
Although formal analysis programming techniques may be quiteold, the introduction of formal methods only dates from the 1980s.These techniques enable us to analyze the behavior of a softwareapplication, described in a programming language. It took until theend of the 1990s before formal methods or the B method could beimplemented in industrial applications or be usable in anindustrial setting. Current literature only gives students and researchers very generaloverviews of formal

methods. The purpose of this book is to
presentfeedback from experience on the use of
"formal methods"(such as proof and model-
checking) in industrial examples withinthe
transportation domain. This book is based on the
experience of people who are currentlyinvolved in
the creation and evaluation of safety critical
systemsoftware. The involvement of people from
within the industry allowsus to avoid the usual
problems of confidentiality which could ariseand thus
enables us to supply new useful information
(photos,architecture plans, real examples, etc.).
Topics covered by the chapters of this book include
SAET-METEOR,the B method and B tools, model-
based design using Simulink, theSimulink design
verifier proof tool, the implementation
andapplications of SCADE (Safety Critical
Application DevelopmentEnvironment), GATeL: A
V&V Platform for SCADE models andControlBuild.
Contents 1. From Classic Languages to Formal
Methods, Jean-LouisBoulanger. 2. Formal Method in
the Railway Sector & the First ComplexApplication:
SAET-METEOR, Jean-Louis Boulanger. 3. The B
Method and B Tools, Jean-Louis Boulanger. 4.
Model-Based Design Using Simulink – Modeling,
CodeGeneration, Verification, and Validation, Mirko
Conrad and PieterJ. Mosterman. 5. Proving Global
Properties with the Aid of the SIMULINK
DESIGNVERIFIER Proof Tool, Véronique Delebarre

andJean-Frédéric Etienne. 6. SCADE: Implementation and Applications, Jean-Louis Camus. 7. GATeL: A V&V Platform for SCADE Models, Bruno Marre,Benjamin Bianc, Patricia Mouy and Christophe Junke. 8. ControlBuild, a Development Framework & for ControlEngineering, Franck Corbier. 9. Conclusion, Jean-Louis Boulanger.

This collection of case studies contains contributions illustrating the application of formal methods to real-life problems with industrial relevance.

Formal methods are mathematically-based techniques, often supported by reasoning tools, that can offer a rigorous and effective way to model, design and analyze computer systems. The purpose of this study is to evaluate international industrial experience in using formal methods. The cases selected are representative of industrial-grade projects and span a variety of application domains. The study had three main objectives: · To better inform deliberations within industry and government on standards and regulations; · To provide an authoritative record on the practical experience of formal methods to date; and À To suggest areas where future research and technology development are needed. This study was undertaken by three experts in formal methods and software engineering: Dan Craigen of ORA Canada, Susan Gerhart of Applied Formal Methods, and Ted Ralston of

Ralston Research Associates. Robin Bloomfield of
Adelard was involved with the Darlington Nuclear
Generating Station Shutdown System case. Support
for this study was provided by organizations in
Canada and the United States. The Atomic Energy
Control Board of Canada (AECB) provided support
for Dan Craigen and for the technical editing
provided by Karen Summerskill. The U.S. Naval
Research Laboratories (NRL), Washington, DC,
provided support for all three authors. The U.S.
National Institute of Standards and Technology
(NIST) provided support for Ted Ralston.
This book constitutes the refereed proceedings of
the 7th International Conference on Formal
Engineering Methods, ICFEM 2005, held in
Manchester, UK in November 2005. The 30 revised
full papers presented together with 3 invited
contributions were carefully reviewed and selected
from 74 submissions. The papers address all current
issues in formal methods and their applications in
software engineering. They are organized in topical
sections on specification, modelling, security,
communication, development, testing, verification,
and tools.
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.

Formal methods for development of computer
systems have been extensively studied over the
years. A range of semantic theories, speci?cation
languages, design techniques, and veri?cation
methods and tools have been developed and applied
to the construction of programs used in critical
applications. The ch- lenge now is to scale up formal
methods and integrate them into engineering -
velopment processes for the correct and e?cient
construction and maintenance of computer systems
in general. This requires us to improve the state of
the art on approaches and techniques for integration
of formal methods into industrial engineering
practice, including new and emerging practice. The
now long-established series of International
Conferences on Formal - gineering Methods brings
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 and to help - vance the state of the art. This
volume contains the papers presented at ICFEM
2009, the 11th International Conference on Formal
Engineering Methods, held during December 9–11,

in Rio de Janeiro, Brazil.
This volume contains the proceedings of the 2003
International Conference on Formal Engineering
Methods (ICFEM 2003). The conference was the
?fth in a series that began in 1997. ICFEM 2003 was
held in Singapore during 5–7 November 2003.
ICFEM 2003 aimed to bring together researchers
and practitioners from - dustry, academia, and
government to advance the state of the art in formal
engineering methods and to encourage a wider
uptake of formal methods in industry. The Program
Committee received 91 submissions from more than
20 co- tries in various regions. After each paper was
reviewed by at least three referees in each relevant
?eld, 34 high-quality papers were accepted based on
originality, technical content, presentation and
relevance to formal methods and software
engineering. We wish to sincerely thank all authors
who submitted their work for consideration. We
would also like to thank the Program Committee
members and other reviewers for their great e?orts
in the reviewing and selecting process. Weareindebt
edtothethreekeynotespeakers,Prof.IanHayesoftheUn
iv- sity of Queensland, Prof. Mathai Joseph of the
Tata Research, Development and DesignCentre,and
Dr.ColinO'HalloranofQinetiQ,foracceptingourinvitatio
n to address the conference.
This book constitutes the refereed proceedings of
the 9th International Conference on Formal

Engineering Methods, ICFEM 2007, held in Boca Raton, Florida, USA, November 14-15, 2007. The 19 revised full papers together with two invited talks presented were carefully reviewed and selected from 38 submissions. The papers address all current issues in formal methods and their applications in software engineering. The papers are organized in topical sections.
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 book provides foundations for software specification and formal software development from the perspective of work on algebraic specification, concentrating on developing basic concepts and studying their fundamental properties. These foundations are built on a solid mathematical basis, using elements of universal algebra, category theory and logic, and this mathematical toolbox provides a convenient language for precisely formulating the concepts involved in software specification and development. Once formally defined, these notions become subject to mathematical

investigation, and this interplay between mathematics and software engineering yields results that are mathematically interesting, conceptually revealing, and practically useful. The theory presented by the authors has its origins in work on algebraic specifications that started in the early 1970s, and their treatment is comprehensive. This book contains five kinds of material: the requisite mathematical foundations; traditional algebraic specifications; elements of the theory of institutions; formal specification and development; and proof methods. While the book is self-contained, mathematical maturity and familiarity with the problems of software engineering is required; and in the examples that directly relate to programming, the authors assume acquaintance with the concepts of functional programming. The book will be of value to researchers and advanced graduate students in the areas of programming and theoretical computer science.
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
This book constitutes the refereed proceedings of the 8th International Conference on Formal Engineering Methods, ICFEM 2006, held in Macao, China, in November 2006. The 38 revised full papers presented together with three keynote talks were carefully reviewed and selected from 108 submissions. The papers address

all current issues in formal methods and their applications in software engineering.

Logic and object-orientation have come to be recognized as being among the most powerful paradigms for modeling information systems. The term "information systems" is used here in a very general context to denote database systems, software development systems, knowledge base systems, proof support systems, distributed systems and reactive systems. One of the most vigorously researched topics common to all information systems is "formal modeling". An elegant high-level abstraction applicable to both application domain and system domain concepts will always lead to a system design from "outside in"; that is, the aggregation of ideas is around real-life objects about which the system is to be designed. Formal methods \yhen applied with this view in mind, especially during early stages of system development, can lead to a formal reasoning on the intended properties, thus revealing system flaws that might otherwise be discovered much later. Logic in different styles and semantics is being used to model databases and their transactions; it is also used to specify concurrent, distributed, real-time, and reactive systems. ,The notion of "object" is central to the modeling of object oriented databases, as well as object-oriented design and programs in software engineering. Both database and software engineering communities have undoubtedly made important contributions to formalisms based on logic and objects. It is worthwhile bringing together the ideas developed by the two communities in isolation, and focusing on integrating

their common strengths.

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.

This text is about the formal specification language Z suitable for courses on Z and formal methods at first and second year undergraduate level. The book includes a tutorial introduction covering the basic mathematics of Z and provides four specification case studies.

Copyright: 19a8b4a32f3ed313c36a72f9eb0f951f