Concur 2009 Concurrency Theory Author Mario Bravetti Sep 2009

HBase: The Definitive GuideResearching Sexual BehaviorMathematical ReviewsC++ Concurrency in ActionInteractive Markov ChainsVerified Software. Theories, Tools, and ExperimentsFundamentals of Computation TheoryFoundations of Software Science and Computation StructuresA Theory of ObjectsC++ Coding StandardsJava Generics and CollectionsVerification of Sequential and Concurrent ProgramsCommunicating Sequential ProcessesTheory and Applications of Models of ComputationProgramming Languages and SystemsReactive SystemsInsider Threats in Cyber SecurityCONCUR 2010 - Concurrency TheoryEmbedded Systems and Software ValidationCONCUR 2013 -- Concurrency TheoryEffective Concurrency in C++Framework Design GuidelinesStructure and Interpretation of Computer Programs - 2nd EditionFoundations of Software Science and Computation StructuresCommunicating and Mobile SystemsCONCUR 2009 - Concurrency TheoryThe Space and Motion of Communicating AgentsScalable PlanningPrinciples of Computer System DesignProgramming Languages and SystemsA Guide to Understanding Covert Channel Analysis of Trusted SystemsCombinatorial Pattern MatchingHandbook of Model CheckingLabelled Markov ProcessesThe Coding Manual for Qualitative ResearchersVerzeichnis lieferbarer Bücher[Books in print / Supplement] ; Books in print : BIP ; an author-title-series index. SupplementFundamentals of Relational Database Management SystemsThe Well-Grounded Java DeveloperSoftware Testing and Analysis

HBase: The Definitive Guide

Provides a set of good practices related to covert channel analysis of systems employed for processing classified and other sensitive information. Written to help vendors and evaluators understand covert channel analysis requirements. Contains suggestions and recommendations. Glossary. References. Illustrations

Researching Sexual Behavior

Concurrency describes any potential time overlap in a set of activities. Its most onerous complexities have been tackled mostly by parallel programmers trying to speed up their applications by harnessing the power of multiple computers (processors, or cores) tied together. Other programmers have mostly remained content with the relative simplicity and ever-increasing speeds of standard sequential (non-parallel) computers, and the rest of us with one-step-at-a-time approaches. But those days are ending: Typical processor speeds have leveled off, and now even laptops and phones are picking up the slack by integrating multiple processors and graphics coprocessors. In the human realm, as communication of all sorts becomes faster and more ubiquitous, we have ever more services (by computers and people) at our disposal, their decentralized nature implying concurrency. How can we humans plan for, and keep track of, all this available concurrency with our "one track" minds? Can these concurrent plans scale up to exploit ever larger collections of processors and/or services? This text uses simple analogies, examples, and thought experiments to explain basic concepts in

concurrency to a broad audience, and to devise an intuitive "elementary particle of activity." A new graphical representation called ScalPL (Scalable Planning Language) is then introduced for building even complex concurrent activities of all kinds from those elemental activities, one mind-sized bite at a time. For programmers, structured and object-oriented programming are extended into the concurrent realm, and performance techniques are explored. For the more serious student, axiomatic semantics and proof techniques are covered. As the world becomes flatter, communication speeds increase, organizations become decentralized, and processors become ubiquitous, Scalable Planning will help you master the trend toward increased concurrency which is here to stay.

Mathematical Reviews

This book constitutes the refereed proceedings of the 18th International Symposium Fundamentals of Computation Theory, FCT 2011, held in Oslo, Norway, in August 2011. The 28 revised full papers presented were carefully reviewed and selected from 78 submissions. FCT 2011 focused on algorithms, formal methods, and emerging fields, such as ad hoc, dynamic and evolving systems; algorithmic game theory; computational biology; foundations of cloud computing and ubiquitous systems; and quantum computation.

C++ Concurrency in Action

Software -- Software Engineering.

Interactive Markov Chains

The Second Edition of Johnny Saldaña's international bestseller provides an in-depth guide to the multiple approaches available for coding qualitative data. Fully up to date, it includes new chapters, more coding techniques and an additional glossary. Clear, practical and authoritative, the book: -describes how coding initiates qualitative data analysis -demonstrates the writing of analytic memos -discusses available analytic software -suggests how best to use The Coding Manual for Qualitative Researchers for particular studies. In total, 32 coding methods are profiled that can be applied to a range of research genres from grounded theory to phenomenology to narrative inquiry. For each approach, Saldaña discusses the method's origins, a description of the method, practical applications, and a clearly illustrated example with analytic follow-up. A unique and invaluable reference for students, teachers, and practitioners of qualitative inquiry, this book is essential reading across the social sciences.

Verified Software. Theories, Tools, and Experiments

This volume contains the proceedings of the 20th Conference on Concurrency Theory (CONCUR 2009), held in Bologna, September 1–4, 2009. The purpose of the CONCUR conference is to bring together researchers, developers, and s-dentsinordertoadvancethetheoryofconcurrencyandpromoteitsapplications. This year the CONCUR conference was in its 20th edition, and to celebrate 20 years of CONCUR, the conference program included a special session organized by the IFIP

Working Groups 1.8 “Concurrency Theory” and 2.2 “Formal - scriptionofProgrammingConcepts”aswellas aninvitedlecturegivenby Robin Milner, one of the fathers of the concurrency theory research area. This edition of the conference attracted 129 submissions. We wish to thank all their authors for their interest in CONCUR 2009. After careful discussions, the Program Committee selected 37 papers for presentation at the conference. Each of them was accurately refereed by at least three reviewers (four reviewers for papers co-authored by members of the Program Committee), who delivered detailedandinsightfulcommentsandsuggestions.TheconferenceChairswarmly thank all the members of the Program Committee and all their sub-referees for the excellent support they gave, as well as for the friendly and constructive discussions. We would also like to thank the authors for having revised their papers to address the comments and suggestions by the referees. The conference program was enriched by the outstanding invited talks by Martin Abadi, Christel Baier, Corrado Priami and, as mentioned above, Robin Milner.

Fundamentals of Computation Theory

Labelled Markov processes are probabilistic versions of labelled transition systems with continuous state spaces. The book covers basic probability and measure theory on continuous state spaces and then develops the theory of LMPs.

Foundations of Software Science and Computation Structures

A Theory of Objects

This book constitutes the thoroughly refereed proceedings of the 24th International Conference on Concurrency Theory, CONCUR 2013, held in Buenos Aires, Argentina, August 27-30, 2013. The 34 revised full papers presented together with 4 invited talks were carefully reviewed and selected from 115 submissions. The papers are organized in topics such as process semantics and modal transition systems, VAS and pushdown systems, Pi calculus and interaction nets, linearizability and verification of concurrent programs, verification of infinite models, model measure and reversibility, stochastic models, message-based interaction processes, principles of automatic verification, and games and control synthesis.

C++ Coding Standards

First account of new theory of communication in computing which describes networks, as well as parts of computer systems.

Java Generics and Collections

This volume constitutes the thoroughly refereed post-conference proceedings of the 8th International Conference on Verified Software: Theories, Tools and Experiments, VSTTE 2016, held in July 2016 in Toronto, ON, Canada. The 8 full papers together with 4 short papers and 5 invited papers presented were carefully

revised and selected 21 submissions. The goal of the VSTTE conference is to advance thestate of the art through the interaction of theory development, tool evolution, and experimental validation.

Verification of Sequential and Concurrent Programs

Insider Threats in Cyber Security is a cutting edge text presenting IT and non-IT facets of insider threats together. This volume brings together a critical mass of well-established worldwide researchers, and provides a unique multidisciplinary overview. Monica van Huystee, Senior Policy Advisor at MCI, Ontario, Canada comments "The book will be a must read, so of course I'll need a copy." Insider Threats in Cyber Security covers all aspects of insider threats, from motivation to mitigation. It includes how to monitor insider threats (and what to monitor for), how to mitigate insider threats, and related topics and case studies. Insider Threats in Cyber Security is intended for a professional audience composed of the military, government policy makers and banking; financing companies focusing on the Secure Cyberspace industry. This book is also suitable for advanced-level students and researchers in computer science as a secondary text or reference book.

Communicating Sequential Processes

Theory and Applications of Models of Computation

By developing object calculi in which objects are treated as primitives, the authors are able to explain both the semantics of objects and their typing rules, and also demonstrate how to develop all of the most important concepts of object-oriented programming languages: self, dynamic dispatch, classes, inheritance, protected and private methods, prototyping, subtyping, covariance and contravariance, and method specialization. An innovative and important approach to the subject for researchers and graduates.

Programming Languages and Systems

Principles of Computer System Design is the first textbook to take a principles-based approach to the computer system design. It identifies, examines, and illustrates fundamental concepts in computer system design that are common across operating systems, networks, database systems, distributed systems, programming languages, software engineering, security, fault tolerance, and architecture. Through carefully analyzed case studies from each of these disciplines, it demonstrates how to apply these concepts to tackle practical system design problems. To support the focus on design, the text identifies and explains abstractions that have proven successful in practice such as remote procedure call, client/service organization, file systems, data integrity, consistency, and authenticated messages. Most computer systems are built using a handful of such abstractions. The text describes how these abstractions are implemented, demonstrates how they are used in different systems, and prepares the reader to apply them in future designs. The book is recommended for junior and senior undergraduate students in Operating Systems, Distributed Systems, Distributed

Operating Systems and/or Computer Systems Design courses; and professional computer systems designers. Features: Concepts of computer system design guided by fundamental principles. Cross-cutting approach that identifies abstractions common to networking, operating systems, transaction systems, distributed systems, architecture, and software engineering. Case studies that make the abstractions real: naming (DNS and the URL); file systems (the UNIX file system); clients and services (NFS); virtualization (virtual machines); scheduling (disk arms); security (TLS). Numerous pseudocode fragments that provide concrete examples of abstract concepts. Extensive support. The authors and MIT OpenCourseWare provide on-line, free of charge, open educational resources, including additional chapters, course syllabi, board layouts and slides, lecture videos, and an archive of lecture schedules, class assignments, and design projects.

Reactive Systems

Structure and Interpretation of Computer Programs by Harold Abelson and Gerald Jay Sussman is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

Insider Threats in Cyber Security

This book constitutes the proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017, which took place in Uppsala, Sweden in April 2017, held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017. The 32 papers presented in this volume were carefully reviewed and selected from 101 submissions. They were organized in topical sections named: coherence spaces and higher-order computation; algebra and coalgebra; games and automata; automata, logic and formal languages; proof theory; probability; concurrency; lambda calculus and constructive proof; and semantics and category theory.

CONCUR 2010 - Concurrency Theory

This book, written by one of the designers of generics, is a thorough explanation of how to use generics, and particularly, the effect this facility has on the way developers use collections.

Embedded Systems and Software Validation

This book provides comprehensive coverage of fundamentals of database management system. It contains a detailed description on Relational Database Management System Concepts. There are a variety of solved examples and review questions with solutions. This book is for those who require a better understanding of relational data modeling, its purpose, its nature, and the standards used in creating relational data model.

CONCUR 2013 -- Concurrency Theory

As multicore and manycore systems become increasingly dominant, handling concurrency will be one of the most crucial challenges developers face. Just as most mainstream programmers have been required to master GUIs and objects, so it will be for concurrency: to achieve the performance they need, developers will have to build and master new libraries, tools, runtime systems, language extensions and above all, new programming best practices. In Effective Concurrency in C++, world-renowned programming guru Herb Sutter identifies and illuminates those best practices. Building on the innovative format pioneered by Scott Meyers's best-selling Effective C++, Sutter presents 35 practical, bite-size chapters, each explaining one proven technique for more successful concurrent programming. Each technique is illuminated through carefully-crafted programming examples written in C++ 0x, the new portable C++ standard - ensuring that programmers will be able to rely on them for many years to come. Sutter also provides case studies and exercises that go beyond the standard "Effective" format to deliver even more engaging hands-on practice, and help developers achieve even deeper mastery.

Effective Concurrency in C++

This book constitutes the proceedings of the 24th European Symposium on Programming, ESOP 2015, which took place in London, UK, in April 2015, held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015. The 33 papers presented in this volume were carefully reviewed and selected from 113 submissions.

Framework Design Guidelines

These are the papers from that conference.

Structure and Interpretation of Computer Programs - 2nd Edition

Markov Chains are widely used as stochastic models to study a broad spectrum of system performance and dependability characteristics. This monograph is devoted to compositional specification and analysis of Markov chains. Based on principles known from process algebra, the author systematically develops an algebra of interactive Markov chains. By presenting a number of distinguishing results, of both theoretical and practical nature, the author substantiates the claim that interactive Markov chains are more than just another formalism: Among other, an algebraic theory of interactive Markov chains is developed, devise algorithms to mechanize compositional aggregation are presented, and state spaces of several million states resulting from the study of an ordinary telefone system are analyzed.

Foundations of Software Science and Computation Structures

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.

Communicating and Mobile Systems

Consistent, high-quality coding standards improve software quality, reduce time-to-market, promote teamwork, eliminate time wasted on inconsequential matters, and simplify maintenance. Now, two of the world's most respected C++ experts distill the rich collective experience of the global C++ community into a set of coding standards that every developer and development team can understand and use as a basis for their own coding standards. The authors cover virtually every facet of C++ programming: design and coding style, functions, operators, class design, inheritance, construction/destruction, copying, assignment, namespaces, modules, templates, genericity, exceptions, STL containers and algorithms, and more. Each standard is described concisely, with practical examples. From type definition to error handling, this book presents C++ best practices, including some that have only recently been identified and standardized-techniques you may not know even if you've used C++ for years. Along the way, you'll find answers to questions like What's worth standardizing--and what isn't? What are the best ways to code for scalability? What are the elements of a rational error handling policy? How (and why) do you avoid unnecessary initialization, cyclic, and definitional dependencies? When (and how) should you use static and dynamic polymorphism together? How do you practice "safe" overriding? When should you provide a no-fail swap? Why and how should you prevent exceptions from propagating across module boundaries? Why shouldn't you write namespace declarations or directives in a header file? Why should you use STL vector and string instead of arrays? How do you choose the right STL search or sort algorithm? What rules should you follow to ensure type-safe code? Whether you're working alone or with others, C++ Coding Standards will help you write cleaner code--and write it faster, with fewer hassles and less frustration.

CONCUR 2009 - Concurrency Theory

C++ Concurrency in Action, Second Edition is the definitive guide to writing elegant multithreaded applications in C++. Updated for C++ 17, it carefully addresses every aspect of concurrent development, from starting new threads to designing fully functional multithreaded algorithms and data structures. Concurrency master Anthony Williams presents examples and practical tasks in every chapter, including insights that will delight even the most experienced developer. -- Provided by publisher.

The Space and Motion of Communicating Agents

If you're looking for a scalable storage solution to accommodate a virtually endless amount of data, this book shows you how Apache HBase can fulfill your needs. As the open source implementation of Google's BigTable architecture, HBase scales to billions of rows and millions of columns, while ensuring that write and read performance remain constant. Many IT executives are asking pointed questions about HBase. This book provides meaningful answers, whether you're evaluating this non-relational database or planning to put it into practice right away. Discover how tight integration with Hadoop makes scalability with HBase easier Distribute large datasets across an inexpensive cluster of commodity servers Access HBase with native Java clients, or with gateway servers providing REST, Avro, or Thrift APIs Get details on HBase's architecture, including the storage format, write-ahead log, background processes, and more Integrate HBase with Hadoop's MapReduce framework for massively parallelized data processing jobs Learn how to tune clusters, design schemas, copy tables, import bulk data, decommission nodes, and many other tasks

Scalable Planning

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

Principles of Computer System Design

Model checking is a computer-assisted method for the analysis of dynamical systems that can be modeled by state-transition systems. Drawing from research traditions in mathematical logic, programming languages, hardware design, and theoretical computer science, model checking is now widely used for the verification of hardware and software in industry. The editors and authors of this handbook are among the world's leading researchers in this domain, and the 32 contributed chapters present a thorough view of the origin, theory, and application of model checking. In particular, the editors classify the advances in this domain and the chapters of the handbook in terms of two recurrent themes that have driven much of the research agenda: the algorithmic challenge, that is, designing model-checking algorithms that scale to real-life problems; and the modeling challenge, that is, extending the formalism beyond Kripke structures and temporal logic. The book will be valuable for researchers and graduate students engaged with the development of formal methods and verification tools.

Programming Languages and Systems

A Guide to Understanding Covert Channel Analysis of Trusted Systems

Summary The Well-Grounded Java Developer offers a fresh and practical look at new Java 7 features, new JVM languages, and the array of supporting technologies you need for the next generation of Java-based software. About the Book The Well-Grounded Java Developer starts with thorough coverage of Java 7 features like try-with-resources and NIO.2. You'll then explore a cross-section of emerging JVM-based languages, including Groovy, Scala, and Clojure. You will find clear examples that are practical and that help you dig into dozens of valuable development techniques showcasing modern approaches to the dev process, concurrency, performance, and much more. Written for readers familiar with Java. No experience with Java 7 or new JVM languages required. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside New Java 7 features Tutorials on Groovy, Scala, and Clojure Discovering multicore processing and concurrency Functional programming with new JVM languages Modern approaches to testing, build, and CI Table of Contents PART 1 DEVELOPING WITH JAVA 7 Introducing Java 7 New I/O PART 2 VITAL TECHNIQUES Dependency Injection Modern concurrency Class files and bytecode Understanding performance tuning PART 3 POLYGLOT PROGRAMMING ON THE JVM Alternative JVM languages Groovy: Java's dynamic friend Scala: powerful and concise Clojure: safer programming PART 4 CRAFTING THE POLYGLOT PROJECT Test-driven development Build and continuous integration Rapid web development Staying well-grounded

Combinatorial Pattern Matching

This open access book constitutes the proceedings of the 23rd International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2020, which took place in Dublin, Ireland, in April 2020, and was held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020. The 31 regular papers presented in this volume were carefully reviewed and selected from 98 submissions. The papers cover topics such as categorical models and logics; language theory, automata, and games; modal, spatial, and temporal logics; type theory and proof theory; concurrency theory and process calculi; rewriting theory; semantics of programming languages; program analysis, correctness, transformation, and verification; logics of programming; software specification and refinement; models of concurrent, reactive, stochastic, distributed, hybrid, and mobile systems; emerging models of computation; logical aspects of computational complexity; models of software security; and logical foundations of data bases.

Handbook of Model Checking

Robin Milner presents a unified structural theory for modelling networks of agents that is destined to have far-reaching significance.

Labelled Markov Processes

This open access book constitutes the proceedings of the 29th European Symposium on Programming, ESOP 2020, which took place in Dublin, Ireland, in April 2020, and was held as Part of the European Joint Conferences on Theory and

Practice of Software, ETAPS 2020. The papers deal with fundamental issues in the specification, design, analysis, and implementation of programming languages and systems. .

The Coding Manual for Qualitative Researchers

The LNCS series reports state-of-the-art results in computer science research, development, and education, at a high level and in both printed and electronic form. Enjoying tight cooperation with the R&D community, with numerous individuals, as well as with prestigious organizations and societies, LNCS has grown into the most comprehensive computer science research forum available. The scope of LNCS, including its subseries LNAI and LNBI, spans the whole range of computer science and information technology including interdisciplinary topics in a variety of application fields. The type of material published traditionally includes -proceedings (published in time for the respective conference) -post-proceedings (consisting of thoroughly revised final full papers) -research monographs (which may be based on outstanding PhD work, research projects, technical reports, etc.) More recently, several color-cover sublines have been added featuring, beyond a collection of papers, various added-value components; these sublines include -tutorials (textbook-like monographs or collections of lectures given at advanced courses) -state-of-the-art surveys (offering complete and mediated coverage of a topic) -hot topics (introducing emergent topics to the broader community)

Verzeichnis lieferbarer Bücher

This is the eBook version of the print title, Framework Design Guidelines, Second Edition . Access to all the samples, applications, and content on the DVD is available through the product catalog page www.informit.com/title/9780321545619 Navigate to the “Downloads” tab and click on the “DVD Contents” links - see instructions in back pages of your eBook. Framework Design Guidelines, Second Edition, teaches developers the best practices for designing reusable libraries for the Microsoft .NET Framework. Expanded and updated for .NET 3.5, this new edition focuses on the design issues that directly affect the programmability of a class library, specifically its publicly accessible APIs. This book can improve the work of any .NET developer producing code that other developers will use. It includes copious annotations to the guidelines by thirty-five prominent architects and practitioners of the .NET Framework, providing a lively discussion of the reasons for the guidelines as well as examples of when to break those guidelines. Microsoft architects Krzysztof Cwalina and Brad Abrams teach framework design from the top down. From their significant combined experience and deep insight, you will learn The general philosophy and fundamental principles of framework design Naming guidelines for the various parts of a framework Guidelines for the design and extending of types and members of types Issues affecting–and guidelines for ensuring–extensibility How (and how not) to design exceptions Guidelines for–and examples of–common framework design patterns Guidelines in this book are presented in four major forms: Do, Consider, Avoid, and Do not. These directives help focus attention on practices that should always be used, those that should generally be used, those that should rarely be used, and those that should never be used. Every guideline includes a discussion of its applicability, and most include a code example to help

illuminate the dialogue. Framework Design Guidelines, Second Edition, is the only definitive source of best practices for managed code API development, direct from the architects themselves. A companion DVD includes the Designing .NET Class Libraries video series, instructional presentations by the authors on design guidelines for developing classes and components that extend the .NET Framework. A sample API specification and other useful resources and tools are also included.

[Books in print / Supplement] ; Books in print : BIP ; an author-title-series index. Supplement

This book constitutes the refereed proceedings of the 20th International Conference on Concurrency Theory, CONCUR 2010, held in Paris, France, August 31 - September 3, 2010. The 35 revised full papers were carefully reviewed and selected from 107 submissions. The topics include: - Basic models of concurrency such as abstract machines, domain theoretic models, game theoretic models, process algebras, and Petri nets. - Logics for concurrency such as modal logics, probabilistic and stochastic logics, temporal logics, and resource logics. - Models of specialized systems such as biology-inspired systems, circuits, hybrid systems, mobile and collaborative systems, multi-core processors, probabilistic systems, real-time systems, service-oriented computing, and synchronous systems. - Verification and analysis techniques for concurrent systems such as abstract interpretation, atomicity checking, model checking, race detection, pre-order and equivalence checking and run-time verification.

Fundamentals of Relational Database Management Systems

The Well-Grounded Java Developer

This book constitutes the refereed proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation, TAMC 2014, held in Singapore, in May 2015. The 35 revised full papers presented were carefully reviewed and selected from 78 submissions. The papers treat all topics relating to the theory and applications of models computation, for example recursion theory and mathematical logic; computational complexity and Boolean functions; graphy theory; quantum computing; parallelism and statistics; learning, automata and probabilistic models; parameterised complexity.

Software Testing and Analysis

Modern embedded systems require high performance, low cost and low power consumption. Such systems typically consist of a heterogeneous collection of processors, specialized memory subsystems, and partially programmable or fixed-function components. This heterogeneity, coupled with issues such as hardware/software partitioning, mapping, scheduling, etc., leads to a large number of design possibilities, making performance debugging and validation of such systems a difficult problem. Embedded systems are used to control safety critical applications such as flight control, automotive electronics and healthcare

monitoring. Clearly, developing reliable software/systems for such applications is of utmost importance. This book describes a host of debugging and verification methods which can help to achieve this goal. Covers the major abstraction levels of embedded systems design, starting from software analysis and micro-architectural modeling, to modeling of resource sharing and communication at the system level Integrates formal techniques of validation for hardware/software with debugging and validation of embedded system design flows Includes practical case studies to answer the questions: does a design meet its requirements, if not, then which parts of the system are responsible for the violation, and once they are identified, then how should the design be suitably modified?