TABLE OF CONTENTS

Preface

Computer software is a persuasive factor in advancing the progress and increasing the efficiency of industry, science, commerce, and communication; it is a significant contributor to the general quality of life of the individual citizen. Future contributions are likely to be even greater. Realization of the potential, and avoidance of the global risks inherent in the introduction of any new technology, will depend on the imagination, inventions, skills, and professional discipline of future cohorts of software engineers. The Marktoberdorf Summer School 2002 on Models, Algebras, and Logic of Engineering Software proved again to deliver excellent education of researchers, teachers, students, and practitioners in the foundations of these disciplines.

As in any engineering discipline, the study and application of the relevant branches of mathematics is crucial in education as well as in practice. It is a prerequisite for the assurance of high reliability and the optimization of cost and benefit of an engineering product. In the case of large-scale software design, the relevant branches of mathematics are logic, algebra, and mathematical modeling in general. The details of the application of theses technologies are increasingly supported by sophisticated toolsets. Therefore, the Marktoberdorf Summer School 2002 was devoted to both mathematical theory and the state of the art of tools.

Algebra

Unifying Models and Engineering Theories of Composed Software Systems
by MANFRED BROY

In general, software is embedded, distributed onto networks, and structured into components that interact, say by message exchange; Manfred Broy lectured on unifying different models and theories of distributed software systems. Various modeling techniques cope separately with various views and elaborate distinct models for data, interfaces, states, and behavioral aspects. The lectures focused on the inter-relationships between different modeling techniques that are to be respected during abstraction, refinement, and composition steps. A meta-model was introduced that specifies consistency requirements among the separate models. The theory introduced in these lectures also reflects diagrammatic description techniques as used in UML and AutoFOCUS, the development tool of Manfred Broy's research group.

Abstract Specification Theory: An Overview
by ANDRZEJ TARLECKI

Abstraction is the essential technique in the control of complexity in software, playing the same simplifying role as approximation in other engineering disciplines. Abstract specification theory was presented by Andrzej Tarlecki. The specifications of realistic systems tend to become very complex, and therefore appropriate techniques are needed which help to control the complexity and develop correct programs. Program development performed by gradual and verifiable refinement steps consequently leads to correct software.

Unifying Theories of Parallel Programming
by JIM WOODCOCK

Combining methods for software engineering was the topic of Jim Woodcock's contribution to the Summer School. He wanted to show how to develop a shared-variable refinement calculus in the style of the sequential calculi of Back, Morgan and Morris. He presented an operational semantics in the style of Hoare and He's unifying theories of programming, which was used to formalize Lamport's Concurrent Hoare Logic. Considering Lamport's Concurrent Hoare Logic, he also gave an invariance proof of a mutual exclusion algorithm as a sample application.

Logic

*Algorithmic and Deductive Verification Methods for CTL**
by AMIR PNUELI

Combining deductive and algorithmic verification methods was treated by Amir Pnueli. The series of lectures gave an introduction to temporal logic and its usage in program specification. It introduced three different kinds of temporal logic (CTL*, CTL, LTL) compared their expressive power and the complexity of model checking problems for different kinds of temporal logic. It also compared two different model checking approaches (state transition graph vs. symbolic) and explained symbolic model checking in detail. At the end it gave an overview of some fundamental results and the corresponding literature.

Software Specification and Verification in Rewriting Logic
by JOSÉ MESEGUER

José Messeguer described experiences in software specification and analysis in rewriting logic. His series of lectures started with the classification of different kinds of software system description: declarative vs. imperative and sequential vs. concurrent. This yields four different system types. Based on this classification José Messeguer introduced different ways of specifying software and showed how to translate these different methods into term rewriting systems. He showed how to prove system properties using these term rewriting rules. Finally, he compared the applicability of theorem proving with model checking for systems with infinite states, yet where the set of reachable states is finite.

1) Exploiting Independence for Verification, Refinement, and Modularity
2) On Translating Models and Properties
by SHMUEL KATZ

Exploiting independence for verification, refinement, and modularity was the aim of Shmuel Katz. His "convenient computation method" reduces the complexity of proving a given property by separating two steps. First, find a subset of special "convenient" computations that satisfy the given property, and then prove that every possible computation is related to such a convenient one in such a way that the relation preserves the property in question. Independence of local operations in different processes can be exploited both during verification of system properties and during system development.

Proving Theorems about Java and the JVM with ACL2
by J. STROTHER MOORE

In Marktoberdorf, theory was confronted with practical needs. An example that has to be mastered is the Java security problem originating in the use of this language for communication on the world-wide net. J. Strother Moore described a methodology for mechanically proving properties of Java methods. An executable model of the JVM formalized in ACL2 allows him to prove theorems about Java methods indirectly by analyzing their representation in Java Bytecode. In various examples including a nontrivial Java method that creates an unbounded number of threads accessing a shared object on the heap, he demonstrated that ACL2 and his formalization of the JVM provide an appropriate level of abstraction for proving theorems about Java and the JVM.

Assertions
by TONY HOARE

Tony Hoare described the evolution of assertions, from early experiences in industry to their role in program proofs today, and how subsequent academic research extended the idea into a methodology for the specification and design of programs. The assertional theory of simple procedural programming is no longer the topic of active research. Rather, there are many aspects of modern programming practice for which adequate assertional techniques and notations are still lacking. Furthermore, the construction of a fully verifying compiler was stated as a long-term challenge for twenty-first century computing science.

Models

From Play-In Scenarios to Code: Capturing and Analyzing Reactive Behavior
by DAVID HAREL

David Harel's course on specifying and executing behavioral requirements was based on the Play-In/Play-Out approach, a powerful setup, within one can conveniently capture scenarios of system behavior, execute them, and simulate the system under development exactly as if it were specified in a conventional state-based fashion. Scenario-based behavior is what practitioners often use when they think about their systems. The lectures showed that it is possible to execute such behavior directly.

Micromodels of Software
by DANIEL JACKSON

Daniel Jackson talked about his tools for declarative modeling and automatic analysis. This series of lectures presented a lightweight first-order specification notation similar to relational algebra. The notation is used to build specifications and generate models satisfying them. The specification is validated manually by generating and treating examples. The lectures explained how the models are generated by translating the specification into a boolean formula and using a SAT-solver to find a model satisfying the formula. He also demonstrated the tool and explained its limitations (such as the undecidability of the specification language and the restriction to finite models). In the last lecture he illustrated his concepts and tool with a case study (bakery algorithm).

Modeling and Verifying a Lego Car Using Hybrid I/O Automata
by FRITS VAANDRAGER

Among the most critical applications of software are timed and hybrid systems. Hybrid systems exhibit discrete as well as continuous behavior. Their verification was the subject of the series of lectures by Frits Vaandrager. He introduced the "Hybrid Input/Output Automaton" (HIOA) modeling framework that is used for the description and analysis of hybrid systems. Inductive methods that are used for proving invariant assertions are extended to the HIOA setting. The timed automata model is introduced as a special subclass of the HIOA model. Industrial case studies illustrate the application of these models.

As can be seen from the foregoing summary, in addition to the importance of its subject matter, a remarkable feature of this Advanced Study Institute was its broad scope, from which its lecturers benefited just as much as the students. The eleven leading experts contributing to this high-quality volume were joined by 72 talented and motivated young scientists from 22 countries. They enjoyed working and living together in the comfortable setting of the Marktoberdorf Gymnasium for eleven days and gained new encouragement and keen insight as well as newly established partnerships.

It is our privilege to thank all those who contributed to the success of this Summer School. These are the lecturers, the participants, our hosts in Marktoberdorf, our staff, our secretary Mrs. INGRID LUHN, and especially our secretary Dr. RALF STEINBRÜGGEN, who organized and accompanied the Summer School for eleven decades. We also thank the publishing assistants from IOS Press for their support.

The Marktoberdorf Summer School was arranged as an Advanced Study Institute of the NATO Science Committee with significant support by the Commission of the European Communities, and with financial aid from the town and district of Marktoberdorf. We thank all authorities involved.

Munich, February 2003 The editors