Lecture Notes in Computer Science 8483

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Marco Bernardo Ferruccio Damiani
Reiner Hähnle Einar Broch Johnsen
Ina Schaefer (Eds.)

Formal Methods for Executable Software Models

14th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2014
Bertinoro, Italy, June 16-20, 2014
Advanced Lectures

Springer

Volume Editors

Marco Bernardo
Università di Urbino, Dipartimento di Scienze di Base e Fondamenti
Piazza della Republica 13, 61029 Urbino, Italy
E-mail: marco.bernardo@uniurb.it

Ferruccio Damiani
Università di Torino, Dipartimento di Informatica
Corso Svizzera 185, 10149 Torino, Italy
E-mail: damiani@di.unito.it

Reiner Hähnle
Technische Universität Darmstadt, Fachbereich Informatik
Hochschulstraße 10, 64289 Darmstadt, Germany
E-mail: haehnle@cs.tu-darmstadt.de

Einar Broch Johnsen
University of Oslo, Department of Informatics
P.O. Box 1080 Blindern, 0316 Oslo, Norway
E-mail: einarj@ifi.uio.no

Ina Schaefer
Technische Universität Braunschweig
Institut für Softwaretechnik und Fahrzeuginformatik
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
E-mail: i.schaefer@tu-braunschweig.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-07316-3 e-ISBN 978-3-319-07317-0
DOI 10.1007/978-3-319-07317-0
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014939047

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume presents a set of papers accompanying the lectures of the 14th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM). This series of schools addresses the use of formal methods in computer science as a prominent approach to the rigorous design of the above-mentioned systems. The main aim of the SFM series is to offer a good spectrum of current research in foundations as well as applications of formal methods, which can be of help for graduate students and young researchers who intend to approach the field. SFM 2014 was devoted to executable software models and covered topics such as variability models, automated analysis techniques, deductive verification, and run-time assessment and testing. The eight papers collected in the two parts of this volume represent the broad range of topics of the school.

The first part is concerned with modeling and verification; it consists of five papers. The paper by Bubel, Flores Montoya, and Hähnle focusses on ABS, the Abstract Behavioral Modeling (ABS) language, and shows how resource consumption analysis, deadlock detection, and functional verification work on ABS models. Giachino and Laneve address recursive programs that admit dynamic resource creation and define a deadlock-detection algorithm based on a generalization to mutations of the theory of permutations of names. The paper by Ábrahám, Becker, Dehnert, Jansen, Katoen, and Wimmer surveys explicit and symbolic techniques for the computation and representation of probabilistic counterexamples for discrete-time Markov chains and probabilistic automata. Gmeiner, Konnov, Schmid, Veith, and Widder illustrate how to integrate parametric data and counter abstraction, finite-state model checking, and abstraction refinement in the setting of threshold-based fault-tolerant distributed algorithms. The paper by Amighi, Blom, Darabi, Huisman, Mostowski, and Zaharieva-Stojanovski discusses the VerCors approach to concurrent software verification, by showing the use of permission-based separation logic to reason about multithreaded Java programs as well as kernel programs following the Single Instruction Multiple Data paradigm.

The second part is on run-time assessment and testing; it contains three papers. De Boer and De Gouw present a method for preventing, isolating, and fixing software bugs, which is based on automated run-time checking of a combination of protocol- and data-oriented properties of object-oriented programs. The paper by Albert, Arenas, Gómez-Zamalloa, and Rojas overviews white-box test-case generation techniques relying on symbolic execution, with emphasis on an implementation in constraint logic programming and an extension to actor-based concurrent software. Finally Lochau, Peldszus, Kowal, and Schaefer describe the activity of model-based testing for single systems and then review techniques

specific to software product lines such as sample-based testing and variability-aware product line testing.

We believe that this book offers a useful view of what has been done and what is going on worldwide in the field of formal methods for executable software models. This school was organized in collaboration with the EU FP7 project Envisage, whose support we gratefully acknowledge. We wish to thank all the speakers and all the participants for a lively and fruitful school. We also wish to thank the entire staff of the University Residential Center of Bertinoro for the organizational and administrative support.

June 2014

Marco Bernardo
Ferruccio Damiani
Reiner Hähnle
Einar Broch Johnsen
Ina Schaefer

Table of Contents

Modeling and Verification

Run-Time Assessment and Testing