Volume Editors

Michael Butler  
University of Southampton  
School of Electronics and Computer Science  
Highfield, Southampton SO17 1BJ, UK  
E-mail: mjb@ecs.soton.ac.uk

Cliff Jones  
Alexander Romanovsky  
Newcastle University  
School of Computing Science  
Newcastle upon Tyne, NE1 7RU, UK  
E-mail: {cliff.jones,alexander.romanovsky}@ncl.ac.uk

Elena Troubitsyna  
Åbo Akademi University  
Department of Computer Science  
Lemminkäisenkatu 14 A, 20520 Turku, Finland  
E-mail: etroubit@abo.fi

Library of Congress Control Number: 2006936100


LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743  

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media  
springer.com  
© Springer-Verlag Berlin Heidelberg 2006  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper  
SPIN: 11916246  06/3142  5 4 3 2 1 0
Foreword

Software is the fuel of the information society. Many of our systems and applications are today controlled and/or developed in software. It is also a well known fact that many software systems have reached a level of complication, mainly because of their size, heterogeneity and distribution (and hopefully not through bad programming), that results in faults appearing which cannot be traced back easily to the code. Some of these “faults” could also be unexpected program behaviour that appears as a result of interactions between the different parts of the program; this is commonly known as complexity. The problem is that sometimes is not easy to say whether a fault is traceable to the code or whether it is due to emergent unexpected behaviour from the complex software system. Testing the code for possible faults is also very costly.

New methods, approaches, tools and techniques are needed to cope with the increasing complexity in software systems; amongst them, fault tolerance techniques and formal methods, supported by the corresponding tools, are promising solutions. This is precisely the subject of this book, which is very much welcome.

The pervasiveness of software in today’s information society makes it of paramount importance, and the main objective of the Software Technologies unit of the European Commission is to support the European software and services industry so that quality software and services are developed to compete in global markets. To help in reaching this objective, it is obvious that we need to maintain and contribute to the excellence in research from universities and research organizations in this specific area.

The volume has been prepared by the partners involved in the FP6 IST-511599 RODIN project (partly funded by the European Commission), “Rigorous Open Development Environment for Complex Systems”. The book brings together papers focusing on the application of rigorous design techniques to the development of fault-tolerant, software-based systems.

In RODIN complexity is mastered by design techniques (specifically formal methods) that support clear thinking and rigorous validation and verification. Coping with complexity also requires architectures that are tolerant of faults and unpredictable changes in the environment; this side is addressed by fault tolerant design techniques. The sources of complexity under study in RODIN are those caused by the environment in which the software is to operate and from the poorly conceived architectural structure.

Who should read this book? Basically, the formal methods and fault tolerance communities. The formal methods people will learn more about (and probably be fired up by) the challenging issues in design for fault tolerance, while researchers on fault tolerance will better understand how formal methods can improve way in which their techniques are developed and applied.

The European Commission, through its successive framework programs, has supported work on methods and techniques to master system complexity and achieve dependable and trustworthy systems. Recently, specifically under the 6th Framework Programme, it has called, amongst other topics, for “Principles, methodologies and tools for design, management and simulation of complex software systems” and
“Foundational and applied research to enable the creation of software systems with properties such as self-adaptability, flexibility, robustness, dependability and evolvability”.

It is clear that these issues are, by no means, fully resolved. Software systems are increasingly complex, and we will need increased efforts in research just to keep up with the pace of development (based on the reflection by the Red Queen in Lewis Carroll's *Through the Looking Glass*, “in this place it takes all the running you can do, to keep in the same place”). It is time, now, for renewed efforts; this book is a pointer in that direction.

August 2006

José-Luis Fernández-Villacañas Martín

Disclaimer: The views expressed in this foreword are those of the author only and should not be construed to reflect or represent the position of the European Commission.
There was, for several decades, a split between researchers who aimed to create perfect programs by using formal methods and those who pioneered techniques for fault tolerance. Of course, the approaches actually complement each other. Fault tolerance generally copes with failures of physical components (and might in specific cases be able to guard against some sorts of design mistakes). Formal reasoning is not just about proving (under assumptions) that a given program will function perfectly; the most productive use of formalism is early on in the design process to help clean up the architecture of a system. As systems have become larger and more intimately linked both to the physical world and to human users, the design task has become far more complex. One of the goals of design must always be to reduce unnecessary complexity in resulting systems.

The editors of this book are proud to be involved in an EU (FP-6) project which specifically brings together researchers from the fault tolerance and formal methods communities. We are aware that through abstraction, refinement and proof, formal methods provide design techniques that support clear thinking as well as rigorous validation and verification. Furthermore, good tool support is essential to support the industrial application of these design techniques.

In 2005 the RODIN (Rigorous Open Development Environment for Complex Systems) project organised a workshop on Rigorous Engineering of Fault Tolerant Systems. REFT 2005\(^1\) was held in conjunction with the Formal Methods 2005 conference at Newcastle University. The aim of this workshop was to bring together researchers who were interested in the application of rigorous design techniques to the development of fault tolerant software based systems.

Such was the success of that event that the organisers decided to prepare a book on the same combination of topics by inviting the authors of the best workshop papers to expand their work and a number of well-established researchers working in the area to write invited chapters. This book contains the refereed and revised papers that came in response. Twelve of the papers are reworked from the workshop; nine of them are totally new. We have also included two provocatively different position statements from Abrial and Amey on the role of programming languages.

The organisers would like to thank the reviewers (some of whom work on RODIN, others are from outside the project): Jean-Raymond Abrial, Elisabeth Ball, Fernando Castor Filho, Patrice Chalin, Ernie Cohen, Joey Coleman, Neil Evans, Massimo Felici, Stefania Gnesi, Stefan Hallerstede, Michael Hansen, Ian Hayes, Alexei Iliasov, Dubravka Ilić, Maciej Koutny, Linas Laibinis, Annabelle McIver, Qaisar Ahmad Malik, César Muñoz, Simin Nadjm-Tehrani, Apostolos Niaouris, Ian Oliver, Patrizio Pelliccione, Mike Poppleton, Shamim Ripon, Colin Snook and Divakar Yadav.

\(^1\) The proceedings are at http://www.cs.ncl.ac.uk/research/pubs/trs/papers/915.pdf
We should particularly like to thank José-Luis Ferández-Villacañas Martin who both gave his time to update the meeting on IST plans and has kindly contributed the Foreword to this volume; and Louise Talbot who has quietly and efficiently handled the collation of this book.

Both in organising REFT 2005 and in publishing this edited book we are aiming to build a network of researchers from the wider community to promote integration of dependability and formal methods research. It is encouraging to see that many of the papers address software based systems that impact peoples’ everyday lives such as communications systems, mobile services, control systems, medical devices and business transactions. We hope that you enjoy reading this volume and encourage you to contribute to our aim of closer collaboration between dependability and formal methods research. We expect to organise another event in London in July 2007: details will appear on the project WWW site http://www.cs.ncl.ac.uk/research/projects/detail.php?id=219

August 2006

Michael Butler
Cliff Jones
Alexander Romanovsky
Elena Troubitsyna
# Table of Contents

Train Systems .................................................... 1  
*Jean-Raymond Abrial*

Formalising Reconciliation in Partitionable Networks with Distributed Services .......................... 37  
*Mikael Asplund, Simin Nadjm-Tehrani*

The Fault-Tolerant Insulin Pump Therapy ......................... 59  
*Alfredo Capozucca, Nicolas Guelfi, Patrizio Pelliccione*

Reasoning About Exception Flow at the Architectural Level ........ 80  
*Fernando Castor Filho, Patrick Henrique da S. Brito, Cecília Mary F. Rubira*

Are Practitioners Writing Contracts? .......................... 100  
*Patrice Chalin*

Determining the Specification of a Control System: An Illustrative Example ............................. 114  
*Joey W. Coleman*

Achieving Fault Tolerance by a Formally Validated Interaction Policy ... 133  
*Alessandro Fantechi, Stefania Gnesi, Laura Semini*

F(I)MEA-Technique of Web Services Analysis and Dependability Ensuring .............................. 153  
*Anatoliy Gorbenko, Vyacheslav Kharchenko, Olga Tarasyuk, Alexey Furmanov*

On Specification and Verification of Location-Based Fault Tolerant Mobile Systems .................. 168  
*Alexei Iliasov, Victor Khomenko, Maciej Koutny, Alexander Romanovsky*

Formal Development of Mechanisms for Tolerating Transient Faults ...... 189  
*Dubravka Ilić, Elena Troubitsyna, Linas Laibinis, Colin Snook*

Separating Concerns in Requirements Analysis: An Example .......... 210  
*Daniel Jackson, Michael Jackson*
Rigorous Fault Tolerance Using Aspects and Formal Methods .......... 226
Shmuel Katz

Rigorous Development of Fault-Tolerant Agent Systems ............... 241
Linas Laibinis, Elena Troubitsyna, Alexei Iliasov,
Alexander Romanovsky

Formal Service-Oriented Development of Fault Tolerant
Communicating Systems .................................................. 261
Linas Laibinis, Elena Troubitsyna, Sari Leppänen, Johan Lilius,
Qaisar Ahmad Malik

Programming-Logic Analysis of Fault Tolerance: Expected Performance
of Self-stabilisation ....................................................... 288
Carroll Morgan, Annabelle K. McIver

Formal Analysis of the Operational Concept for the Small Aircraft
Transportation System .................................................. 306
César Muñoz, Víctor Carreño, Gilles Dowek

Towards a Method for Rigorous Development of Generic Requirements
Patterns ................................................................. 326
Colin Snook, Michael Poppleton, Ian Johnson

Rigorous Design of Fault-Tolerant Transactions for Replicated
Database Systems Using Event B ...................................... 343
Divakar Yadav, Michael Butler

Engineering Reconfigurable Distributed Software Systems: Issues
Arising for Pervasive Computing ...................................... 364
Apostolos Zarras, Manel Fredj, Nikolaos Georgantas, Valerie Issarny

Position Papers

Tools for Developing Large Systems (A Proposal) ..................... 387
Jean-Raymond Abrial

Why Programming Languages Still Matter .......................... 391
Peter Amey

Author Index .............................................................. 403