Jim Davies
Wolfram Schulte
Mike Barnett (Eds.)

LNCS 3308

Formal Methods and Software Engineering

6th International Conference on Formal Engineering Methods, ICFEM 2004 Seattle, WA, USA, November 2004 Proceedings

Springer

Jim Davies Wolfram Schulte
Mike Barnett (Eds.)

Formal Methods and Software Engineering

6th International Conference
on Formal Engineering Methods, ICFEM 2004
Seattle, WA, USA, November 8-12, 2004
Proceedings

Springer

Volume Editors

Jim Davies
University of Oxford, Software Engineering Programme
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
E-mail: jim.davies@comlab.ox.ac.uk

Wolfram Schulte
Mike Barnett
Microsoft Research
One Microsoft Way, Cedar Court 113/4048, Redmond, WA 98052-6399, USA
E-mail: {schulte, mbarnett}@microsoft.com

Library of Congress Control Number: 2004114617

CR Subject Classification (1998): D.2.4, D.2, D.3, F.3

ISSN 0302-9743
ISBN 3-540-23841-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11348801 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3308

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

Editorial Board

Lecture Notes in Computer Science

For information about Vols. 1–3203

please contact your bookseller or Springer

Vol. 3308: J. Davies, W. Schulte, M. Barnett (Eds.), Formal Methods and Software Engineering. XIII, 500 pages. 2004.

Vol. 3305: P.M.A. Sloot, B. Chopard, A.G. Hoekstra (Eds.), Cellular Automata. XV, 883 pages. 2004.

Vol. 3302: W.-N. Chin (Ed.), Programming Languages and Systems. XIII, 453 pages. 2004.

Vol. 3299: F. Wang (Ed.), Automated Technology for Verification and Analysis. XII, 506 pages. 2004.

Vol. 3295: P. Markopoulos, B. Eggen, E. Aarts, J.L. Crowley (Eds.), Ambient Intelligence. XIII, 388 pages. 2004.

Vol. 3294: C.N. Dean, R.T. Boute (Eds.), Teaching Formal Methods. X, 249 pages. 2004.

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3292: R. Meersman, Z. Tari, A. Corsaro (Eds.), On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. XXIII, 885 pages. 2004.

Vol. 3291: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 824 pages. 2004.

Vol. 3290: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 823 pages. 2004.

Vol. 3289: S. Wang, K. Tanaka, S. Zhou, T.W. Ling, J. Guan, D. Yang, F. Grandi, E. Mangina, I.-Y. Song, H.C. Mayr (Eds.), Conceptual Modeling for Advanced Application Domains. XXII, 692 pages. 2004.

Vol. 3288: P. Atzeni, W. Chu, H. Lu, S. Zhou, T.W. Ling (Eds.), Conceptual Modeling – ER 2004. XXI, 869 pages. 2004.

Vol. 3287: A. Sanfeliu, J.F. Martínez Trinidad, J.A. Carrasco Ochoa (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XVII, 703 pages. 2004.

Vol. 3286: G. Karsai, E. Visser (Eds.), Generative Programming and Component Engineering. XIII, 491 pages. 2004.

Vol. 3285: S. Manandhar, J. Austin, U. Desai, Y. Oyanagi, A. Talukder (Eds.), Applied Computing. XII, 334 pages. 2004.

Vol. 3284: A. Karmouch, L. Korba, E.R.M. Madeira (Eds.), Mobility Aware Technologies and Applications. XII, 382 pages. 2004.

Vol. 3281: T. Dingsøyr (Ed.), Software Process Improvement. X, 207 pages. 2004.

Vol. 3280: C. Aykanat, T. Dayar, İ. Körpeoğlu (Eds.), Computer and Information Sciences - ISCIS 2004. XVIII, 1009 pages. 2004.

Vol. 3278: A. Sahai, F. Wu (Eds.), Utility Computing. XI, 272 pages. 2004.

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. Lopez, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3252: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004 Workshops. XVIII, 785 pages. 2004.

Vol. 3251: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004. XXII, 1025 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Vol. 3246: A. Apostolico, M. Melucci (Eds.), String Processing and Information Retrieval. XIV, 332 pages. 2004.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004. (Subseries LNAI).

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004. (Subseries LNAI).

Vol. 3243: S. Leonardi (Ed.), Algorithms and Models for the Web-Graph. VIII, 189 pages. 2004.

Vol. 3242: X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII. XX, 1185 pages. 2004.

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3236: M. Núñez, Z. Maamar, F.L. Pelayo, K. Pousttchi, F. Rubio (Eds.), Applying Formal Methods: Testing, Performance, and M/E-Commerce. XI, 381 pages. 2004.

Vol. 3235: D. de Frutos-Escrig, M. Nunez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004. X, 377 pages. 2004.

Vol. 3234: M.J. Egenhofer, C. Freksa, H.J. Miller (Eds.), Geographic Information Science. VIII, 345 pages. 2004.

Vol. 3233: K. Futatsugi, F. Mizoguchi, N. Yonezaki (Eds.), Software Security - Theories and Systems. X, 345 pages. 2004.

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3231: H.-A. Jacobsen (Ed.), Middleware 2004. XV, 514 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M. Saiz Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004. (Subseries LNAI).

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004. (Subseries LNAI).

Vol. 3226: M. Bouzeghoub, C. Goble, V. Kashyap, S. Spaccapietra (Eds.), Semantics of a Networked World. XIII, 326 pages. 2004.

Vol. 3225: K. Zhang, Y. Zheng (Eds.), Information Security. XII, 442 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logics. VIII, 337 pages. 2004.

Vol. 3222: H. Jin, G.R. Gao, Z. Xu, H. Chen (Eds.), Network and Parallel Computing. XX, 694 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3219: M. Heisel, P. Liggesmeyer, S. Wittmann (Eds.), Computer Safety, Reliability, and Security. XI, 339 pages. 2004.

Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 1114 pages. 2004.

Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 930 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVII, 906 pages. 2004. (Subseries LNAI).

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1302 pages. 2004. (Subseries LNAI).

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1280 pages. 2004. (Subseries LNAI).

Vol. 3212: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 862 pages. 2004.

Vol. 3211: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 880 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004. (Subseries LNAI).

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3204: C.A. Peña Reyes, Coevolutionary Fuzzy Modeling. XIII, 129 pages. 2004.

Preface

Formal engineering methods are changing the way that software systems are developed. With language and tool support, they are being used for automatic code generation, and for the automatic abstraction and checking of implementations. In the future, they will be used at every stage of development: requirements, specification, 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, and to help advance the state of the art. Authors are strongly encouraged to make their ideas as accessible as possible, and there is a clear emphasis upon work that promises to bring practical, tangible benefit: 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 first to be held in North America. Previous conferences were held in Singapore, China, UK, Australia, and Japan. The Programme Committee received 110 papers and selected 30 for presentation. The final versions of those papers are included here, together with 2-page abstracts for the 5 accepted tutorials, and shorter abstracts for the 4 invited talks.

We would like to thank: Dines Bjørner, for his work in organizing speakers and sponsors; Jin Song Dong and Jim Woodcock, for an excellent handover from ICFEM 2003; Joxan Jaffar, J Strother Moore, Peter Neumann, and Amitabh Srivastava, for agreeing to address the conference; the authors, for submitting their work; the Programme Committee, and their colleagues, for their reviews; and Springer, for their help with publication.

ICFEM 2004 was organized by Microsoft Research in Seattle, with additional support and sponsorship from the University of Oxford, the United Nations University, Formal Methods Europe, NASA, and ORA Canada.

November 2004

Jim Davies
Wolfram Schulte
Mike Barnett

Organizing Committee

Conference Committee

Mike Barnett (Microsoft Research, USA)
Local Organization
Dines Bjørner (National University of Singapore, Singapore)
Conference Chair
Jim Davies (University of Oxford, UK)
Programme Co-chair
Wolfram Schulte (Microsoft Research, USA)
Programme Co-chair
Hongjun Zheng (Semantics Design, USA)
Workshops and Tutorials Chair

Sponsors

Microsoft Research
`www.research.microsoft.com`
Oxford University Software Engineering Programme
`www.softeng.ox.ac.uk`
United Nations University–International Institute for Software Technology
`www.iist.unu.edu`
Formal Methods Europe (FME)
`www.fmeurope.org`
NASA–JPL Laboratory for Reliable Software
`eis.jpl.nasa.gov/lars/`
ORA Canada
`www.ora.on.ca`

Steering Committee

Keijiro Araki (Kyushu University, Japan)
Jin Song Dong (National University of Singapore, Singapore)
Chris George (United Nations University, Macau)
Jifeng He *(Chair)* (IIST, United Nations University, Macau)
Mike Hinchey (NASA, USA)
Shaoying Liu (Hosei University, Japan)
John McDermid (University of York, UK)
Tetsuo Tamai (University of Tokyo, Japan)
Jim Woodcock (University of York, UK)

Programme Committee

Adnan Aziz (University of Texas, USA)
Richard Banach (University of Manchester, UK)
Egon Börger (University of Pisa, Italy)
Jonathan Bowen (London South Bank University, UK)
Manfred Broy (University of Munich, Germany)
Michael Butler (University of Southampton, UK)
Ana Cavalcanti (University of Kent, UK)
Dan Craigen (ORA, Canada)
Jin Song Dong (National University of Singapore, Singapore)
Matthew Dwyer (Kansas State University, USA)
John Fitzgerald (University of Newcastle upon Tyne, UK)
David Garlan (Carnegie Mellon University, Pittsburgh, USA)
Thomas Jensen (IRISA/CNRS Campus de Beaulieu, Rennes, France)
Jim Larus (Microsoft Research, USA)
Mark Lawford (McMaster University, Canada)
Huimin Lin (Chinese Academy of Sciences, Beijing, China)
Peter Lindsay (University of Queensland, Australia)
Shaoying Liu (Hosei University, Japan)
Zhiming Liu (United Nations University, Macau SAR, China)
Brendan Mahony (Department of Defence, Australia)
Marc Frappier (Université de Sherbrooke, Québec, Canada)
William Bradley Martin (National Security Agency, USA)
David Notkin (University of Washington, USA)
Jeff Offutt (George Mason University, USA)
Harald Ruess (Computer Science Laboratory, SRI, USA)
Augusto Sampaio (Universidade Federal de Pernambuco, Brazil)
Thomas Santen (Technical University of Berlin, Germany)
Doug Smith (Kestrel Institute, USA)
Graeme Smith (University of Queensland, Australia)
Paul A. Swatman (University of South Australia, Australia)
Sofiene Tahar (Concordia University, Canada)
T.H. Tse (Hong Kong University, Hong Kong)
Yi Wang (Uppsala University, Sweden)
Farn Wang (National Taiwan University, Taiwan)
Jeannette Wing (Carnegie Mellon University, USA)
Jim Woodcock (University of York, UK)

Reviewers

Amr Abdel-Hamid; Isaam Al-azzoni; Adnan Aziz; Richard Banach; Andreas Bauer; Jounaidi Ben Hassen; Egon Börger; Jonathan Bowen; Peter Braun; Manfred Broy; Michael Butler; Colin Campbell; Ana Cavalcanti; Alessandra Cavarra; Antonio Cerone; Yifeng Chen; Corina Cirstea; David Clark; Dan Craigen; Charles Crichton; Jim Davies; Roger Duke; Bruno Dutertre; Matthew Dwyer; Pao-Ann Eric Hsiung; Yuan Fang Li; Bill Farmer; William M. Farmer; Carla Ferreira; Colin Fidge; John Fitzgerald; Marc Frappier; Jorn Freiheit; David Garlan; Amjad Gawanmeh; Frederic Gervais; Jeremy Gibbons; Uwe Glaesser; Andy Gravell; Wolfgang Grieskamp; Ali Habibi; John Hakansson; Steve Harris; Jifeng He; Maritta Heisel; Steffen Helke; Matthew Hennessy; Xiayong Hu; Geng-Dian Huang; Chung-Yang Ric Huang; Van Hung Dang; Jiale Huo; Cornelia P. Inggs; Jan Jürjens; Bart Jacobs; Thomas Jensen; Thierry Jeron; Zhi Jin; Wolfram Kahl; Soon-Kyeong Kim; Soon Kyeong Kim; Leonid Kof; Pushmeet Kohli; Pavel Krcal; Sy-Yen Kuo; Rom Langerak; James Larus; Mark Lawford; Ryan Leduc; Karl Lermer; Guangyuan Li; Xiaoshan Li; Huimin Lin; Peter Lindsay; Shaoying Liu; Zhiming Liu; Quan Long; Marcio Lopes Cornelio; Dorel Lucanu; Anthony MacDonald; Brendan Mahony; William Bradley Martin; Jim McCarthy; M. Meisinger; Yassine Mokhtari; Leonid Mokrushin; Alexandre Mota; Muan Yong Ng; Sidney Nogueira; David Notkin; Jeff Offutt; Chun Ouyang; Hong Pan; Jun Pang; Paul Pettersson; Mike Poppleton; Steven Postma; Stephane Lo Presti; Wolfgang Reisig; Abdolbaghi Rezazadeh; River; Harald Ruess; Heinrich Rust; Vlad Rusu; Augusto Sampaio; Thomas Santen; Renate Schmidt; Wolfram Schulte; Thorsten Schutt; Dirk Seifert; Laura Semini; Adnan Sherif; Benjamin Sigonneau; Carlo Simon; Andrew Simpson; Doug Smith; Graeme Smith; Doug Smith; Colin Snook; Jin Song Dong; Maria Sorea; Mark Staples; Jun Sun; Paul A. Swatman; Sofiene Tahar; J.P. Talpin; Rodrigo Teixeira Ramos; Nikolai Tillmann; T.H. Tse; Phillip J. Turner; Margus Veanes; S. Vogel; Philip Wadler; Farn Wang; Bow-Yaw Wang; Alan Wassyng; Jun Wei; Guido Wimmel; Jeannette Wing; Kirsten Winter; Jim Woodcock; Wang Yi; Fang Yu; Mohamed Zaki; Wenhui Zhang; Guangquan Zhang; Ning Zhang; Riley Zheng; Xiaocong Zhou; Jeff Zucker

Table of Contents

Tutorials

Invited Talks

Full Papers

Model-Based Development: Combining Engineering Approaches and Formal Techniques

Bernhard Schätz

Institut für Informatik, Technische Universität München,
80290 München, Germany

1 Model-Based Development

In a nutshell, a model-based approach offers several benefits:

Improved Product: By moving away from an implementation biased view of a system, the developer can focus on the important issues of the product under development. This results in
- thinking in terms of the domain-specific conceptual model (state, interaction, etc.) instead of the coding level (objects, method calls, etc.)
- narrowing the gap between informal specification and formal specification (since, e.g., notions like mode, event, or communication, appear in the informal specifications as well as in the model)
- limiting the possibility of making mistakes while building models and refining them by ensuring consistency conditions (e.g. interface correctness, absence of undefined behavior)

Improved Process: Using a better structured product model helps to identify more defects earlier. Additionally, higher efficiency can be obtained by more CASE-supported process steps
- mechanizing conceptual consistency conditions, either guaranteed by construction (e.g., interface correctness) or checked automatically on demand (e.g., completeness of defined behavior)
- supporting semantical consistency conditions, either automatically (e.g., checking whether an EET can be performed by system), or interactively (e.g., by ensuring a safety condition of a system)
- enabling transformations of specification (e.g. inserting standard behavior for undefined situations), interactively carried out by the CASE tool.

The limit of model-based support is defined by the sophistication of the underlying model: by adding more domain-specific aspects, more advanced techniques can be offered (e.g., reliability analysis, schedule generation).

2 Structured Development

Model-based development is all about adding the structuring and preciseness of formal approaches to the development process while being driven by the models of the application domain instead of mathematical formalisms.

J. Davies et al. (Eds.): ICFEM 2004, LNCS 3308, pp. 1–2, 2004.

Thus it helps reducing the complexity of the development process by clearly focusing on specific issues and offering suitable, generally CASE-supported techniques tailored for each step of the development (see also [1]):

Step 1: By linking informal requirements and the models of system and environment, we can check whether all informal requirements are covered or trace back from design to requirements; but most importantly, it helps to structure the requirements, get a better understanding, and identify open issues in the requirements.

Step 2: By modeling the environment and its interface to the system, we get a precise understanding of how the system influences the environment and how the environment reacts. Especially important, we precisely state what properties of the environment we take for granted and which possibly faulty behavior we must tolerate to establish the required safety behavior.

Step 3: By defining the abstract, possibly incomplete and non-deterministic behavior of the system, we describe a high-level initial design without already incorporating implementation details complicating the design or limiting further decision. Furthermore, we establish the basis for further analysis and development steps.

Step 4: By analyzing the models of system and environment for their validity, we ensure their usefulness for the following steps; using simulation or conceptual consistency checks, we identify open issues like unrealistic behavior of the environment, as well as undefined behavior of the system.

Step 5: By using even stronger techniques we establish whether the defined behavior of the system ensures the expected behavior of the environment. Using formal verification, we show that the system guarantees the safety requirements of the environment assuming optimal circumstances; furthermore, using failure analysis, we show that also mean time requirements are met using quantitative assumptions about the environment.

Step 6: By applying structural refinement, we add further design decisions concerning the architecture of a system by breaking it up into interacting sub-components. By adding behavior to each sub-component, we can make sure that the refined system implements the abstract one.

Step 7: By restricting non-deterministic or undefined behavior, we add further design decisions concerning underspecified behavior of the introduced components, resulting in a more optimized reaction of the system while maintaining the overall functionality.

Step 8: By checking the components of the system for possible undefined behavior we identify remaining open behavior to be resolved in the final implementation. By adding standard reactions for open behavior, we improve the robustness of the system against unexpected behavior.

References

1. B. Schätz. Mastering the Complexity of Embedded Systems - The AutoFocus Approach. In F. Kordon and M. Lemoine, editors, *Formal Techniques for Embedded Distributed Systems: From Requirements to Detailed Design.* Kluwer, 2004.

Tutorial on the RAISE Language, Method and Tools

Chris George

United Nations University, International Institute for Software Technology (UNU-IIST), Macao
cwg@iist.unu.edu
http://www.iist.unu/~cwg

Abstract. RAISE — Rigorous Approach to Industrial Software Engineering — was first developed in European collaborative projects during 1985-94. Since then a new set of free, open-source, portable tools has been developed, and a range of developments have been carried out. This tutorial introduces the language and method, and demonstrates the range of software life-cycle activities supported by the tools. These include generation of specifications from UML class diagrams, validation and verification of specifications, refinement, prototyping, execution of test cases, mutation testing, generation of documents, proof by means of translation to PVS, and generation of program code in C++ by translation. A comprehensive user guide is also available.

It is a common perception that "formal methods" are difficult to use, involve a lot of proof, are expensive, and are only applicable to small, critical problems. The aim of this tutorial is to introduce a formal technology that is easy to learn, need involve no proof at all, is cheap to apply, and is intended in particular to deal with large, not necessarily critical problems.

The RAISE method [1] is extreme in its basic simplicity: you write a specification and then you produce the implementation (if possible, automatically from the specification). You may for a particular project want to do more than this: you may decide you want to produce a more abstract specification first, and then a more concrete one, and then perhaps assert (and even prove) some relation, such as refinement, between them. But for many projects one specification is enough [2]. If you find graphical approaches useful, you can start by drawing a UML class diagram and generate the first specification from it.

What this simple picture of the method hides is the urgent need to validate the specification. Many discussions on formal methods assume the specification is correct, is what you want, and concentrate on verification, on showing that it is properly implemented. But all this is wasted if the specification is wrong. Mistakes made at the beginning of a project (and not quickly noticed) cause many problems, at best time and cost overruns, and at worst complete failure. So a major aim of the RAISE method, and of the tools that support it, is exploring and finding problems with the specification.

J. Davies et al. (Eds.): ICFEM 2004, LNCS 3308, pp. 3–4, 2004.

Concretely we can, with the tools:

- write large specifications in modules that we can analyse separately;
- generate and inspect "confidence conditions" that subtype conditions are not violated, pre-conditions are satisfied by function calls, cases are complete, etc.;
- translate and execute test cases that are recorded with the specification;
- assess the adequacy of test cases by means of specification mutation testing;
- generate high quality documents that include the specifications; and even, for more critical applications,
- prove confidence conditions, refinement, or other properties that we choose to assert

The RAISE Specification Language (RSL) [3] is a "wide-spectrum" language. It supports "algebraic" as well as "model-oriented" styles, and also includes applicative, imperative and concurrent features. It is modular, and therefore supports the writing of a very wide range of specifications.

Introductory information on the language and method is available in a variety of papers [4–6]. There is also an extension to RSL to deal with time [7].

The RAISE tool is open source, comes ready built for Windows, Linux and Sparc-Solaris, and can be built in any environment where you can compile C. There is also a comprehensive user guide [8].

The tutorial covers RSL, the method, the tools, and gives an example of a large system specified using RAISE.

References

1. The RAISE Language Group. *The RAISE Development Method.* BCS Practitioner Series. Prentice Hall, 1995. Available from `ftp://ftp.iist.unu.edu/pub/RAISE/method_book`.
2. Hung Dang Van, Chris George, Tomasz Janowski, and Richard Moore. *Specification Case Studies in RAISE.* FACIT. Springer-Verlag, 2002.
3. The RAISE Language Group. *The RAISE Specification Language.* BCS Practitioner Series. Prentice Hall, 1992. Available from Terma A/S. Contact `jnp@terma.com`.
4. Chris George. A RAISE Tutorial. Technical Report 153, UNU-IIST, P.O.Box 3058, Macau, December 1998. Presented at the BRNS workshop *Verification of Digital and Hybrid Systems* at TIFR, Mumbai, India, 7–11 January 1999.
5. Chris George. Introduction to RAISE. Technical Report 249, UNU-IIST, P.O. Box 3058, Macau, April 2002.
6. Chris George and Anne E. Haxthausen. The Logic of the RAISE Specification Language. *Computing and Informatics*, 22(3–4), 2003.
7. Chris George and Xia Yong. An Operational Semantics for Timed RAISE. Technical Report 149, UNU-IIST, P.O.Box 3058, Macau, November 1998. Presented at and published in the proceedings of FM'99, Toulouse, France, 20–24 September 1999, LNCS 1709, Springer-Verlag, 1999, pp. 1008–1027.
8. Chris George. RAISE Tools User Guide. Technical Report 227, UNU-IIST, P.O. Box 3058, Macau, February 2001. The tools are available from `http://www.iist.unu.edu`.