Judi Romijn
Graeme Smith
Jaco van de Pol (Eds.)

LNCS 3771

Integrated Formal Methods

5th International Conference, IFM 2005
Eindhoven, The Netherlands, November/December 2005
Proceedings

Judi Romijn Graeme Smith
Jaco van de Pol (Eds.)

Integrated Formal Methods

5th International Conference, IFM 2005
Eindhoven, The Netherlands
November 29 – December 2, 2005
Proceedings

Springer

Volume Editors

J.M.T. Romijn
Eindhoven University of Technology, Computing Science Department
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: jromijn@win.tue.nl

G.P. Smith
The University of Queensland
School of Information Technology and Electrical Engineering
4072 Australia
E-mail: smith@itee.uq.edu.au

J.C. van de Pol
Centre for Mathematics and Computer Science (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
E-mail: Jaco.van.de.Pol@cwi.nl

Library of Congress Control Number: 2005935883

CR Subject Classification (1998): F.3, D.3, D.2, D.1

ISSN	0302-9743
ISBN-10	3-540-30492-4 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-30492-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11589976 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3771

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

Editorial Board

Preface

This is the 5th edition of the International Conference on Integrated Formal Methods (IFM). Previous IFM conferences were held in York (June 1999), Dagstuhl (November 2000), Turku (May 2002) and Canterbury (April 2004). This year's IFM was held in December 2005 on the campus of the Technische Universiteit Eindhoven in The Netherlands.

This year IFM received 40 submissions, from which 19 high-quality papers were selected by the Program Committee. Besides these, the proceedings contain invited contributions by Patrice Godefroid, David Parnas and Doron Peled.

It was 10 years ago that Jonathan P. Bowen and Michael G. Hinchey published their famous *Ten Commandments of Formal Methods* in IEEE Computer 28(4). Their very first commandment — *Thou shalt choose an appropriate notation* — touches the heart of the IFM theme: Complex systems have different aspects, and each aspect requires its own appropriate notation.

Classical examples of models for various aspects are: state based notations and algebraic data types for data, process algebras and temporal logics for behavior, duration calculus and timed automata for timing aspects, etc. The central question is how the models of different notations relate. Recently, Bowen and Hinchey presented their *Ten Commandments Revisited* (in: ACM proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems). They distinghuish variations in combining notations, ranging from *loosely coupled viewpoints* to *integrated methods.*

The loosely coupled viewpoints are quite popular (cf. the success of UML) and are easy to adopt in a leightweight process. They could be useful for specifying and analyzing isolated system aspects. However, the main advantage of formal methods — being able to specify and verify the correctness of complete systems — is lost.

In order to specify and verify complete systems, an integrated approach is inescapable. Integrated methods provide an underlying concept, a semantic integration of models, an integrated methodology and integrated verification algorithms. The added value is that questions regarding inter-model consistency, completeness and correctness of implementations become meaningful and can be answered effectively. Bowen and Hinchey acknowledge this as the central theme of the series of IFM conferences.

These proceedings contain new insights in the field of integrated formal methods. The various papers contribute to integration at notational, semantic and tool level. We hope that the reader will find inspiring material and useful knowledge. Ultimately, we hope that the field contributes to more reliable software and hardware systems, constructed with less effort.

We would like to thank all PC members and all anonymous referees for their excellent and timely job in assessing the quality of the submitted papers. We

also thank the invited speakers for their contributions. We are grateful to Holger Hermanns for his invited tutorial on QoS modelling and analysis for embedded systems. Finally, we thank our home institutes Technische Universiteit Eindhoven, University of Queensland and CWI for their support.

September 2005 Judi Romijn, Graeme Smith, Jaco van de Pol

Program Committee

Didier Bert (France)
Eerke Boiten (UK)
Jonathan Bowen (UK)
Michael Butler (UK)
Paul Curzon (UK)
Jim Davies (UK)
John Derrick (UK)
Steve Dunne (UK)
Jin Song Dong (Singapore)
Andy Galloway (UK)
Chris George (Macau, SAR China)
Wolfgang Grieskamp (USA)
Henri Habrias (France)
Maritta Heisel (Germany)
Soon-Kyeong Kim (Australia)
Michel Lemoine (France)
Shaoying Liu (Japan)
Dominique Mery (France)
Stephan Merz (France)
Richard Paige (UK)
Luigia Petre (Finland)
Jaco van de Pol
(Co-chair, The Netherlands)
Judi Romijn
(Co-chair, The Netherlands)
Thomas Santen (Germany)
Steve Schneider (UK)
Wolfram Schulte (USA)
Kaisa Sere (Finland)
Jane Sinclair (UK)
Graeme Smith
(Co-chair, Australia)
Bill Stoddart (UK)
Kenji Taguchi (Japan)
Helen Treharne (UK)
Heike Wehrheim (Germany)
Kirsten Winter (Australia)
Jim Woodcock (UK)

Sponsors

We thank NWO and FME for sponsoring the invited lectures, IPA for sponsoring the welcome reception, and BCS-FACS for sponsoring the best paper awards.

External Referees

Besides the Program Committee members, several external anonymous referees read the submitted papers. Without their help, the conference would have been of a lesser quality. The following is a list, to the best of our knowledge, of external referees.

Bernhard Aichernig
Pascal André
Victor Bos
Chunqing Chen
Neil Evans
David Faitelson
Lars Grunske
Ping Hao
Bart Jacobs
Hironobu Kuruma
Alistair McEwan
Sotiris Moschoyiannis
Stephane Lo Presti
Ivan Porres Paltor
Jean-Claude Reynaud
Rimvydas Ruksenas
Colin Snook
Yasuyuki Tahara
Nikolai Tillmann
Leonidas Tsiopoulos
G. Vidal-Naquet
James Welch
Hirokazu Yatsu

Lecture Notes in Computer Science

For information about Vols. 1–3698

please contact your bookseller or Springer

Vol. 3807: M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, Q.Z. Sheng (Eds.), Web Information Systems – WISE 2005 Workshops. XV, 275 pages. 2005.

Vol. 3806: A.H. H. Ngu, M. Kitsuregawa, E.J. Neuhold, J.-Y. Chung, Q.Z. Sheng (Eds.), Web Information Systems Engineering – WISE 2005. XXI, 771 pages. 2005.

Vol. 3799: A. Rodríguez, I.F. Cruz, S. Levashkin (Eds.), GeoSpatial Semantics. X, 259 pages. 2005.

Vol. 3793: T. Conte, N. Navarro, W.-m.W. Hwu, M. Valero, T. Ungerer (Eds.), High Performance Embedded Architectures and Compilers. XIII, 317 pages. 2005.

Vol. 3792: I. Richardson, P. Abrahamsson, R. Messnarz (Eds.), Software Process Improvement. VIII, 215 pages. 2005.

Vol. 3791: A. Adi, S. Stoutenburg, S. Tabet (Eds.), Rules and Rule Markup Languages for the Semantic Web. X, 225 pages. 2005.

Vol. 3790: G. Alonso (Ed.), Middleware 2005. XIII, 443 pages. 2005.

Vol. 3789: A. Gelbukh, Á. de Albornoz, H. Terashima-Marín (Eds.), MICAI 2005: Advances in Artificial Intelligence. XXVI, 1198 pages. 2005. (Subseries LNAI).

Vol. 3785: K.-K. Lau, R. Banach (Eds.), Formal Methods and Software Engineering. XIV, 496 pages. 2005.

Vol. 3784: J. Tao, T. Tan, R.W. Picard (Eds.), Affective Computing and Intelligent Interaction. XIX, 1008 pages. 2005.

Vol. 3781: S.Z. Li, Z. Sun, T. Tan, S. Pankanti, G. Chollet, D. Zhang (Eds.), Advances in Biometric Person Authentication. XI, 250 pages. 2005.

Vol. 3780: K. Yi (Ed.), Programming Languages and Systems. XI, 435 pages. 2005.

Vol. 3779: H. Jin, D. Reed, W. Jiang (Eds.), Network and Parallel Computing. XV, 513 pages. 2005.

Vol. 3777: O.B. Lupanov, O.M. Kasim-Zade, A.V. Chaskin, K. Steinhöfel (Eds.), Stochastic Algorithms: Foundations and Applications. VIII, 239 pages. 2005.

Vol. 3775: J. Schönwälder, J. Serrat (Eds.), Ambient Networks. XIII, 281 pages. 2005.

Vol. 3773: A.S. Cortés, M.L. Cortés (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XX, 1094 pages. 2005.

Vol. 3772: M. Consens, G. Navarro (Eds.), String Processing and Information Retrieval. XIV, 406 pages. 2005.

Vol. 3771: J.M.T. Romijn, G.P. Smith, J. van de Pol (Eds.), Integrated Formal Methods. XI, 407 pages. 2005.

Vol. 3770: J. Akoka, S.W. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. van den Heuvel, M. Kolp, J.C. Trujillo, C. Kop, H.C. Mayr (Eds.), Perspectives in Conceptual Modeling. XXII, 476 pages. 2005.

Vol. 3768: Y.-S. Ho, H.J. Kim (Eds.), Advances in Mulitmedia Information Processing - PCM 2005, Part II. XXVIII, 1088 pages. 2005.

Vol. 3767: Y.-S. Ho, H.J. Kim (Eds.), Advances in Mulitmedia Information Processing - PCM 2005, Part I. XXVIII, 1022 pages. 2005.

Vol. 3766: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 231 pages. 2005.

Vol. 3765: Y. Liu, T. Jiang, C. Zhang (Eds.), Computer Vision for Biomedical Image Applications. X, 563 pages. 2005.

Vol. 3764: S. Tixeuil, T. Herman (Eds.), Self-Stabilizing Systems. VIII, 229 pages. 2005.

Vol. 3762: R. Meersman, Z. Tari, P. Herrero (Eds.), On the Move to Meaningful Internet Systems 2005: OTM Workshops. XXXI, 1228 pages. 2005.

Vol. 3761: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Part II. XXVII, 653 pages. 2005.

Vol. 3760: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Part I. XXVII, 921 pages. 2005.

Vol. 3759: G. Chen, Y. Pan, M. Guo, J. Lu (Eds.), Parallel and Distributed Processing and Applications - ISPA 2005 Workshops. XIII, 669 pages. 2005.

Vol. 3758: Y. Pan, D. Chen, M. Guo, J. Cao, J. Dongarra (Eds.), Parallel and Distributed Processing and Applications. XXIII, 1162 pages. 2005.

Vol. 3757: A. Rangarajan, B. Vemuri, A.L. Yuille (Eds.), Energy Minimization Methods in Computer Vision and Pattern Recognition. XII, 666 pages. 2005.

Vol. 3756: J. Cao, W. Nejdl, M. Xu (Eds.), Advanced Parallel Processing Technologies. XIV, 526 pages. 2005.

Vol. 3754: J. Dalmau Royo, G. Hasegawa (Eds.), Management of Multimedia Networks and Services. XII, 384 pages. 2005.

Vol. 3753: O.F. Olsen, L.M.J. Florack, A. Kuijper (Eds.), Deep Structure, Singularities, and Computer Vision. X, 259 pages. 2005.

Vol. 3752: N. Paragios, O. Faugeras, T. Chan, C. Schnörr (Eds.), Variational, Geometric, and Level Set Methods in Computer Vision. XI, 369 pages. 2005.

Vol. 3751: T. Magedanz, E.R. M. Madeira, P. Dini (Eds.), Operations and Management in IP-Based Networks. X, 213 pages. 2005.

Vol. 3750: J.S. Duncan, G. Gerig (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005, Part II. XL, 1018 pages. 2005.

Vol. 3749: J.S. Duncan, G. Gerig (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005, Part I. XXXIX, 942 pages. 2005.

Vol. 3748: A. Hartman, D. Kreische (Eds.), Model Driven Architecture – Foundations and Applications. IX, 349 pages. 2005.

Vol. 3747: C.A. Maziero, J.G. Silva, A.M.S. Andrade, F.M.d. Assis Silva (Eds.), Dependable Computing. XV, 267 pages. 2005.

Vol. 3746: P. Bozanis, E.N. Houstis (Eds.), Advances in Informatics. XIX, 879 pages. 2005.

Vol. 3745: J.L. Oliveira, V. Maojo, F. Martín-Sánchez, A.S. Pereira (Eds.), Biological and Medical Data Analysis. XII, 422 pages. 2005. (Subseries LNBI).

Vol. 3744: T. Magedanz, A. Karmouch, S. Pierre, I. Venieris (Eds.), Mobility Aware Technologies and Applications. XIV, 418 pages. 2005.

Vol. 3740: T. Srikanthan, J. Xue, C.-H. Chang (Eds.), Advances in Computer Systems Architecture. XVII, 833 pages. 2005.

Vol. 3739: W. Fan, Z.-h. Wu, J. Yang (Eds.), Advances in Web-Age Information Management. XXIV, 930 pages. 2005.

Vol. 3738: V.R. Syrotiuk, E. Chávez (Eds.), Ad-Hoc, Mobile, and Wireless Networks. XI, 360 pages. 2005.

Vol. 3735: A. Hoffmann, H. Motoda, T. Scheffer (Eds.), Discovery Science. XVI, 400 pages. 2005. (Subseries LNAI).

Vol. 3734: S. Jain, H.U. Simon, E. Tomita (Eds.), Algorithmic Learning Theory. XII, 490 pages. 2005. (Subseries LNAI).

Vol. 3733: P. Yolum, T. Güngör, F. Gürgen, C. Özturan (Eds.), Computer and Information Sciences - ISCIS 2005. XXI, 973 pages. 2005.

Vol. 3731: F. Wang (Ed.), Formal Techniques for Networked and Distributed Systems - FORTE 2005. XII, 558 pages. 2005.

Vol. 3729: Y. Gil, E. Motta, V. R. Benjamins, M.A. Musen (Eds.), The Semantic Web – ISWC 2005. XXIII, 1073 pages. 2005.

Vol. 3728: V. Paliouras, J. Vounckx, D. Verkest (Eds.), Integrated Circuit and System Design. XV, 753 pages. 2005.

Vol. 3726: L.T. Yang, O.F. Rana, B. Di Martino, J. Dongarra (Eds.), High Performance Computing and Communications. XXVI, 1116 pages. 2005.

Vol. 3725: D. Borrione, W. Paul (Eds.), Correct Hardware Design and Verification Methods. XII, 412 pages. 2005.

Vol. 3724: P. Fraigniaud (Ed.), Distributed Computing. XIV, 520 pages. 2005.

Vol. 3723: W. Zhao, S. Gong, X. Tang (Eds.), Analysis and Modelling of Faces and Gestures. XI, 4234 pages. 2005.

Vol. 3722: D. Van Hung, M. Wirsing (Eds.), Theoretical Aspects of Computing – ICTAC 2005. XIV, 614 pages. 2005.

Vol. 3721: A. Jorge, L. Torgo, P.B. Brazdil, R. Camacho, J. Gama (Eds.), Knowledge Discovery in Databases: PKDD 2005. XXIII, 719 pages. 2005. (Subseries LNAI).

Vol. 3720: J. Gama, R. Camacho, P.B. Brazdil, A. Jorge, L. Torgo (Eds.), Machine Learning: ECML 2005. XXIII, 769 pages. 2005. (Subseries LNAI).

Vol. 3719: M. Hobbs, A.M. Goscinski, W. Zhou (Eds.), Distributed and Parallel Computing. XI, 448 pages. 2005.

Vol. 3718: V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing. XII, 502 pages. 2005.

Vol. 3717: B. Gramlich (Ed.), Frontiers of Combining Systems. X, 321 pages. 2005. (Subseries LNAI).

Vol. 3716: L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, Ó. Pastor (Eds.), Conceptual Modeling – ER 2005. XVI, 498 pages. 2005.

Vol. 3715: E. Dawson, S. Vaudenay (Eds.), Progress in Cryptology – Mycrypt 2005. XI, 329 pages. 2005.

Vol. 3714: H. Obbink, K. Pohl (Eds.), Software Product Lines. XIII, 235 pages. 2005.

Vol. 3713: L.C. Briand, C. Williams (Eds.), Model Driven Engineering Languages and Systems. XV, 722 pages. 2005.

Vol. 3712: R. Reussner, J. Mayer, J.A. Stafford, S. Overhage, S. Becker, P.J. Schroeder (Eds.), Quality of Software Architectures and Software Quality. XIII, 289 pages. 2005.

Vol. 3711: F. Kishino, Y. Kitamura, H. Kato, N. Nagata (Eds.), Entertainment Computing - ICEC 2005. XXIV, 540 pages. 2005.

Vol. 3710: M. Barni, I. Cox, T. Kalker, H.J. Kim (Eds.), Digital Watermarking. XII, 485 pages. 2005.

Vol. 3709: P. van Beek (Ed.), Principles and Practice of Constraint Programming - CP 2005. XX, 887 pages. 2005.

Vol. 3708: J. Blanc-Talon, W. Philips, D.C. Popescu, P. Scheunders (Eds.), Advanced Concepts for Intelligent Vision Systems. XXII, 725 pages. 2005.

Vol. 3707: D.A. Peled, Y.-K. Tsay (Eds.), Automated Technology for Verification and Analysis. XII, 506 pages. 2005.

Vol. 3706: H. Fuks, S. Lukosch, A.C. Salgado (Eds.), Groupware: Design, Implementation, and Use. XII, 378 pages. 2005.

Vol. 3704: M. De Gregorio, V. Di Maio, M. Frucci, C. Musio (Eds.), Brain, Vision, and Artificial Intelligence. XV, 556 pages. 2005.

Vol. 3703: F. Fages, S. Soliman (Eds.), Principles and Practice of Semantic Web Reasoning. VIII, 163 pages. 2005.

Vol. 3702: B. Beckert (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods. XIII, 343 pages. 2005. (Subseries LNAI).

Vol. 3701: M. Coppo, E. Lodi, G. M. Pinna (Eds.), Theoretical Computer Science. XI, 411 pages. 2005.

Vol. 3700: J.F. Peters, A. Skowron (Eds.), Transactions on Rough Sets IV. X, 375 pages. 2005.

Vol. 3699: C.S. Calude, M.J. Dinneen, G. Păun, M. J. Pérez-Jiménez, G. Rozenberg (Eds.), Unconventional Computation. XI, 267 pages. 2005.

Table of Contents

Invited Papers

Session: Components

Session: State/Event-Based Verification

Session: System Development

Session: Applications of B

Session: Tool Support

Session: Non-software Domains

Session: Semantics

Session: UML and Statecharts

A Family of Mathematical Methods for Professional Software Documentation

David Lorge Parnas

Software Quality Research Laboratory (SQRL),
Department of Computer Science and Information Systems,
Faculty of Informatics and Electronics,
University of Limerick, Limerick, Ireland

1 Introduction

The movement to integrate mathematically based software development methods is a predictable response to the fact that none of the many methods available seems sufficient to do the whole job (whatever that may be) on its own. This talk argues that integrating separately developed methods is not the most fruitful possible approach. Instead we propose a family of methods, based on a common model, designed to be complementary and mutually supportive.

The method family being developed at the Software Quality Research Lab at the University of Limerick is characterised by two major decisions:

- Software developers must prepare and maintain a set of documents whose content (not format) is specified by the relational model presented in [3].
- The relations are represented using mathematical expressions in tabular form. [5].

This talk will motivate these decisions, describe the model, illustrate the concept of tabular expressions, and discuss the uses of such documents in software development.

2 Why We Need Better Software Development Documentation

Software has its, well earned, bad reputation in large part because developers do not provide an appropriate set of design documentation. When developers are trying to extend, correct, or interface with a piece of software, they need detailed information that is both precise and correct. This information is usually hard to find, missing, imprecisely expressed, or simply wrong. Some developers estimate that they spend 80% of their time seeking information about existing software. When they don't have an interface description, they often use implementation information that is subject to change. The result is software in which small changes have surprising effects.

Engineers who design physical products are taught how to use mathematics to specify properties of their products and analyse their designs. Software developers simply do not know how to do that and their managers do not expect them to do it. In fact, the profession doesn't know how to do it. Finding ways to use mathematics to produce precise, well-organized development documentation should be a major research effort.

J. Romijn, G. Smith, and J. van de Pol (Eds.): IFM 2005, LNCS 3771, pp. 1 – 4, 2005.

3 Why Models are not Generally Descriptions

It has become fashionable to propose the use of "model based development" (MBT) to solve some of these problems. Formal looking[1] models are prepared to express design decisions. Unfortunately, these models do not constitute documentation; to understand why, it is important to distinguish between descriptions, specifications, and models.

- A *description* of a product provides accurate information about that product.
- A *specification* is a description that states all of the requirements but contains no other information.
- A *model* has some of the properties of the product-of-interest but may also have properties that differ from those of the product. Some models are themselves products, but here we are concerned primarily with mathematical models.

We use the word "*document*" to mean either a specification or another description.

It is important to note that the above are statements of intent, most attempts at descriptions are not completely accurate and it is very difficult to write a complete specification (using any method).

- A description of a product provides information about that product. A single description need not be complete; we can use a collection of descriptions. However, everything that the description states must be true of the product, else the description is wrong. A description is only useful if it is easier to derive information from the description than to derive it from the product itself.
- If a specification is correct but not an accurate description of a product, it is the product that is wrong. If the specification is true of a product, but the product is not satisfactory, the specification must be wrong (either inaccurate or just incomplete). If the specification is not true of a product, but the product is satisfactory, the specification is deficient (either an overspecification or incorrect).
- A model is something with some of the properties of a product; however, not all of the model's properties need be true of the product itself. A model may not be a specification or a description. Models are potentially dangerous because one can derive information from a model that is not true of the product. All information that is based on a model must be taken "with a grain of salt".

4 No New Mathematics

For the work described in the remainder of the talk, there is no new mathematics; we have merely found new ways to apply classical mathematical concepts. This is common in Engineering research.

The logic used is very close to the classical logic presented in textbooks. Our only deviation from the most basic logic is that the meaning of predicates involving partial functions is fully defined (predicates are total). [2]

We make heavy use of the concepts of function (including functions of time) and relation. We represent functions in the traditional way, using mathematical expressions.

[1] The meaning of these models is often not precisely defined.

Relations that are not functions are represented by giving the characteristic predicate of the set of ordered pairs, again using a mathematical expression. The use of the relational model in [3] is the most basic innovation in our work.

The only mathematics that may appear new is our use of a multi-dimensional format for expression, which we call tabular expressions. These are no more "powerful" than traditional expressions; in fact, we define their meaning by telling how to construct an equivalent traditional expression. The advantage is purely notational; For the class of functions/relations of interest to us, the tabular form of the expression is usually easier to construct, check and read. The talk provides numerous illustrations of this notation.

5 Need for Document Definitions

When industrial developers take the time to write a basic document such as a requirements document, there are often debates about what should be in it. While, there is a plethora of standards in the software field, those standards detail format and structure but not contents. As a result, even documents that satisfy standards, usually lack essential information. The standards provide no way of saying that a document is complete. When we had produced the first such [1], there was widespread agreement that it was an unusually useful document, but strong disagreement about which standard it satisfied. In contrast, many documents that fully satisfied the standards that were in place were not found useful because they did not contain essential information.

In [1] we have proposed definitions of the following documents:

1. "System Requirements Document"- Treats computer systems as "black-box".
2. "System Design Document"- Describes computers and communication.
3. "Software Requirements Document" - Describes required software behaviour
4. "Software Function Specification": Describes actual software behaviour.
5. "Software Module Guide": How to find your module. (informal document).
6. "Module Interface Specifications": Treats each module as black-box.
7. "Uses Relation Document": Range and domain comprise module programs.
8. "Module Internal Design Documents": Describe data structure and programs.

For each of the above except 5, which is informal, we have stated what variables should be defined in the document and what relations must be represented/defined by the document. These definitions provide an unambiguous basis for deciding what should be contained in a document. Documents may still be incomplete, but we can identify the missing information and check for important properties.

6 Tabular Notation

Although conventional mathematical notation is capable of describing the functions and relations called for by our definitions, many find those expressions very hard to parse. Documents written using conventional notation are precise but difficult to review and not very useful as reference documents. This is because the nature of the

functions that we encounter in software. Traditional engineering mathematics stresses functions that are continuous and differentiable. With digital computers we implement functions that approximate piecewise-continuous functions. Through experience we have discovered that multi-dimensional forms of expressions are well suited to this class of functions. Many practical applications have shown that tabular expressions are better accepted, more easily prepared, more easily checked, and more useful than the conventional expressions that they replace. Of course, conventional expressions are still essential: (1) the innermost cells of a tabular expression always contain conventional expressions and (2) our new approach to defining the meaning of these expressions shows how to transform it to an equivalent conventional expression.

The last point is important. Many who see these expressions find them simpler and easier to read than "formal method" examples and assume that they are somehow less formal than other methods. Formality need not mean "hard to read". Tabular expressions are just as formal as the conventional expressions that they replace.

7 Practical Advantages of Mathematics

The fact that our documents are based on conventional mathematics has a great many practical advantages. It allows us to use existing theorem proving programs to check tables for consistency and completeness, to use computer algebra systems to transform and simplify documents, to evaluate expressions when simulating something that we have specified, to generate test oracles from specifications, and to generate test cases based on black box documents. We have also found these documents extremely valuable for inspecting critical software. [4]

8 Future Work

Based on our new, much more general (yet simpler) definition of tabular expressions, we are building a new generation of tools to support the use of this approach.

References

[1] Heninger, K., Kallander, J., Parnas, D.L., Shore, J., "Software Requirements for the A-7E Aircraft", NRL Report 3876, November 1978, 523 pgs.

[2] Parnas, D.L., "Predicate Logic for Software Engineering", *IEEE Transactions on Software Engineering*, Vol. 19, No. 9, September 1993, pp. 856 - 862 Reprinted as Chapter 3 in [6]

[3] Parnas, D.L., Madey, J., "Functional Documentation for Computer Systems Engineering" *Science of Computer Programming* (Elsevier) vol. 25, no. 1, Oct. 1995, pp 41-61

[4] Parnas, D.L. "Inspection of Safety Critical Software using Function Tables", *Proceedings of IFIP World Congress 1994, Volume III"* August 1994, pp. 270 - 277. Also Ch. 19 in [6]

[5] Janicki, R., Parnas, D.L., Zucker, J., "Tabular Representations in Relational Documents", Chapter 12 in "*Relational Methods in Computer Science*", Ed. C. Brink and G. Schmidt. Springer Verlag, pp. 184 - 196, 1997, ISBN 3-211-82971-7. Reprinted as Chapter 4 in [6].

[6] Hoffman, D.M., Weiss, D.M. (eds.), "*Software Fundamentals: Collected Papers by David L. Parnas*", Addison-Wesley, 2001, 664 pgs., ISBN 0-201-70369-6.

Generating Path Conditions for Timed Systems

Saddek Bensalem[1], Doron Peled[2,*], Hongyang Qu[2], and Stavros Tripakis[1]

[1] Verimag, 2 Avenue de Vignate, 38610 Gieres, France
[2] Department of Computer Science, University of Warwick, Coventry, CV4 7AL United Kingdom

Abstract. We provide an automatic method for calculating the path condition for programs with real time constraints. This method can be used for the semiautomatic verification of a unit of code in isolation, i.e., without providing the exact values of parameters with which it is called. Our method can also be used for the automatic generation of test cases for unit testing. The current generalization of the calculation of path condition for the timed case turns out to be quite tricky, since not only the selected path contributes to the path condition, but also the timing constraints of alternative choices in the code.

1 Introduction

Software testing often involves the use of informal intuition and reasoning. But it is possible to employ some formal methods techniques and provide tools to support it. Such tools can help in translating the informal ideas and intuition into formal specification, assist in searching the code, support the process of inspecting it and help analyzing the results. A tester may have a vague idea where problems in the code may occur. The generation of a condition for a generated suspicious sequence may help the tester to confirm or refute such a suspicion. Such a condition relates the variables at the beginning of the sequence. Starting the execution with values satisfying this condition is necessary to recreate the execution.

We generalize the calculation of a path condition, taking into account only the essential conditions to follow a particular path in the execution. We start with a given path merely from practical consideration; it is simpler to choose a sequence of program statements to execute. However, we look at the essential partial order, which is consistent with the real-time constraints, rather than at the total order. We cannot assume that transitions must follow each other, unless this order stems from some sequentiality constraints such as transitions belonging to the same process or using the same variable or from timing constraints.

For untimed systems, there is no difference between the condition for the partial order execution and the condition to execute any of the sequences (linearizations) consistent with it. Because of commutativity between concurrently

* This research was partially supported by Subcontract UTA03-031 to The University of Warwick under University of Texas at Austin's prime National Science Foundation Grant #CCR-0205483.

J. Romijn, G. Smith, and J. van de Pol (Eds.): IFM 2005, LNCS 3771, pp. 5–19, 2005.