Int J Software Informatics, Vol.3, No.2-3, June/September 2009, pp. 121–127 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
 Tel: +86-10-62661040

Editorial

Dines Bjørner

(DTU Informatics, The Technical University of Denmark, DK-2800 Kgs.Lyngby, Denmark)

Bjørner D. Editorial. *Int J Software Informatics*, **2009, 3(2-3): 121–127. http://www.ijsi.org/1673-7288/3/121.htm**

1 Programming Methodology

1.1 Delineations

Computer science is the study and knowledge of the kind of "things" that can "exist inside" computers and communication. *Computing science* is the study and knowledge of how to construct the artifacts that can "exist inside" computers and communication. *Programming methodology* is part of computing science. *Software engineering* is applied computing science. By a *method* we shall understand a set of *principles* for `selecting` and `applying` a number of `analysis` and `synthesis` *techniques* and *tools* in the construction of an artifact — here software. The *principles* (for `selection` and `application`) are necessarily to be ultimately carried out by humans: new software artifacts are, in all interesting cases, of a complexity where automation of `selection` and `application` is not possible; usually the predicates determining `selection` and `application` are not computable. We therefore prefer to use the double term *formal techniques* rather than *formal methods* (one cannot formalise the `selection` and `application` *principles*).

The above delineations focus only on the topics covered by the papers of this double issue.

1.2 History of formal specification

The quest for *formal programming methodology techniques* is motivated by the need to develop trustworthy software: software that is right, that is, can be proven, checked and/or formally tested correct, and software that is the right software, that is, fulfills customers/users expectations.

Since the late 1960s *formal programming methodology techniques* have evolved. It started with Tony Hoare's *The Axiomatic Basis of Computer Programming*[1]. It continued, notably, with the `Vienna Development Method` [Papers 7 and 8], `VDM`, emerging in the early 1970s at the IBM Vienna Laboratory[2−4]; notable `VDM` R&D co-workers were (the late) Hans Bekič, Dines Bjørner, Peter Lucas and Cliff Jones. Jean-Raymond Abrial first contributed, around 1980 with `Z`[5−7] [Paper 9], around 1990 with `B`[8], and around 2000 with `Event B`[9] [Paper 4]. `RAISE`[10−14] [Paper 5] (rigorous approach to industrial software engineering) with its specification language `RSL` emerged during the late 1980s; notable `RAISE` R&D co-workers were Chris George and

(the late) Søren Prehn. LOTOS [Paper 10], a formal specification technique for specifying concurrent and distributed systems, emerged around the mid-to-late 1980s[16]; notable originator of LOTOS was Ed Brinksma. Duration Calculus, DC [Paper 3], a method and a specification language for dealing with dense time temporal notions developed as a result of the ProCoS project[17,18]; notable DC R&D co-workers, in the 1990s and into this century were and are Zhou Chao Chen and Michael Reichhardt Hansen. ASM (abstract state machines)[19] emerged from evolving algebras in the mid 1990s and originated with Yuri Gurevich [Paper 2]. Alloy[20] [Paper 1], the last of the formal techniques approaches illustrated in this double-issue, emerged in the early 2000s. Alloy was developed by Daniel Jackson and his students. Alloy R&D is now spreading beyond MIT. But formal development of correct software need not only be done within a (set of) formal specification language(s). Good, old, reliable mathematics [Paper 6] is still very much a proposition – as illustrated in Paper 6[1].

I expect that our readers will appreciate this double issue as it focuses on the careful development of provably correct software from abstract specifications.

2 Special Double-Issue Papers

2.1 Listing of papers

1. **E. Kang, D. Jackson** :
   Designing and Analyzing a Flash System with **Alloy** (**Alloy**)
2. **M. Veanes, N. Bjørner, Y. Gurevich, W. Schulte** :
   Symbolic Bounded Model Checking of **Abstract State Machines** (**ASM: Model Checking**)
3. **M. Fränzle, M.R. Hansen** :
   Efficient Model Checking for **Duration Calculus** (**DC**)
4. **D. Méry** :
   Refinement-Based Guidelines for Algorithmic Systems (**Event B**)
5. **A. Haxthausen** :
   Developing a Domain Model for Relay Circuits (**RAISE**)
6. **B. Charron-Bost and S. Merz** :
   Formal Verification of a Consensus Algorithm in the Heard-Of Model (**Mathematics**)
7. **P.G. Larsen, J. Fitzgerald, S.Wolff** :
   Methods for the Development of Distributed Real-Time Embedded Systems using **VDM** (**VDM**)
8. **T. Kurita, Y. Nakatsugawa** :
   The Application of **VDM** to the Industrial Development of Firmware for a Smart Card IC Chip (**VDM**)
9. **L. Freitas and J.C.P. Woodcock** :
   A Chain Datatype in **Z** (**Z**)

[1]The exciting, German VeriSoft project [http://www.verisoft.de], headed by Wolfgang Paul, applies mathematics more-or-less throughout: From gate circuit hardware development via computer architecture, operating system, compiler and other base systems development to application software development.

10. M. Weiglhofer, B. Aichernig, F. Wotawa :
Fault-Based Conformance Testing in Practice **(LOTOS: Testing)**

2.2 Paper selection criteria

The papers for this double-issue of IJSI were solicited such that the above leading formal and model-oriented techniques were all represented. One (paper # 2) emphasizes model-checking (rather than 'using' ASM in some development) and one paper (# 10) was solicited to cover formal testing.

2.3 Brief classification

In the previous paragraph we put most of the papers of this double issue in the historical context of the R&D of formal techniques. Below we enumerate these papers in the alphabetic order of the prime name of the formal technique they espouse. Papers 1., 4., 5., 6., 7. and 9. very explicitly develop an abstract, formal specification into a concrete, likewise formal specification, typically in the same specification language (or "in" mathematics), but such that the concrete specification resembles that of a program in some typical programing language. Whereas the abstract, formal specification expresses properties, the concrete program "implements" those properties. Paper 8. reports on a rather large scale, and rather successful industrial development, from abstract, formal to concrete formal, yet executable VDM specification. Papers 2. and 3. are concerned with the model checking of properties of abstract specifications (for ASM and DC). Paper 10. reports on formal testing of LOTOS specifications. Together these 10 papers show that formal techniques in software engineering has reached a mature stage and are industrially applicable — while still providing fertile ground for further programming methodology research. All papers clearly illustrate the expression of theorems and proofs of correctness and all papers clearly illustrate the use of and/or need for software tools.

3 Acknowledgements

The editor and IJSI thank all authors for their hard work. From my initial invitation in mid October 2008 via their delivery of papers for review by mid March 2009 and their kind reactions to extensive reviews, the authors, listed above, have delivered. The editor and IJSI also thank reviewers. A few have asked to remain anonymous. The rest are listed here in alphabetic order:

- Keijiro Araki
- Jens Bendisposto
- J. C. Bicarregui
- Nikolaj Bjørner
- Jonathan P. Bowen
- Michael Butler
- Andrew Butterfield
- Chunqing Chen
- Lars-Henrik Eriksson
- John Fitzgerald
- Martin Fränzle
- Chris W. George
- Michael R. Hansen
- Klaus Havelund
- Martin Henson
- Rob Hierons
- Thierry Jéron
- Peter Gorm Larsen
- Ben Moszkowski
- Paritosh K. Pandya
- Steve Reeves
- Elvinia Riccobene
- Marcel Verhoef
- Shaofa Yang

Several referees reviewed two or three papers. All papers were refereed by at least three reviewers. In most cases their referees' reports were very detailed and thus not only helped the authors and thus the readers, you, but also brought computing science several steps further. Again, the IJSI and I cannot over-emphasize their contributions: very many and dear thanks !

The editor thanks the IJSI Editor-in-Chief and the IJSI Associate Editors-in-Chief for their gracious support in promoting this double-issue. The editor and the authors of this double-issue thank the issue copy-editor, Ms. Shi Lihong, for her outstanding work in preparing the manuscripts for final type-setting and for communicating promptly and patiently with all of us.

4 Bibliographical Notes

This double-issue does not cover all, but I believe important, approaches to formal development of software. Below we summarise a more complete list of current approaches. Notable additions are those which cover diagrammatic approaches ($\mathcal{D}$), algebraic approaches ($\mathcal{A}$) and additional temporal logic approaches ($\mathcal{T}$): `Alloy`[20], `ASM`[19], `B`[8], `Event B`[9], `CafeOBJ` ($\mathcal{A}$)[21,22], `CASL` ($\mathcal{A}$)[23,24], `CSP`[25−28], `Duration Calculus` ($\mathcal{T}$)[17,18] (`DC`), `Live Sequence Charts` ($\mathcal{D}$)[29] (`LSC`), `Message Sequence Charts` ($\mathcal{D}$)[30−32] (`MSC`), `RAISE`[10−14] (`RSL`), `Petri nets` ($\mathcal{D}$)[33−35], `Statecharts` ($\mathcal{D}$)[36], `Temporal Logic of Reactive Systems` ($\mathcal{T}$)[37−40] (a basis for the Stanford Temporal Prover[41,42]), `TLA+` ($\mathcal{T}$)[43,44] (`Temporal Logic of Actions`), `VDM`[2−4], and `Z`[5−7]. The recent monograph on *Logics of Specification Languages*[45] covers `ASM, Event B, CafeObj, CASL, DC, RAISE, TLA+, VDM` and `Z`.

References

[1] Hoare CAR. The axiomatic basis of computer programming. Communications of the ACM, 1969, 12(10): 567–583.

[2] Bjørner D, Jones CB, eds. The Vienna Development Method: The Meta-Language, Lecture Notes in Computer Science,Springer, 1978. 61.

[3] Bjørner D, Jones CB, eds. Formal Specification and Software Development. Prentice Hall, Hemel Hampstead, England, 1982.

[4] Fitzgerald JS, Larsen PG. Modelling Systems: Practical Tools and Techniques in Software Development. Cambridge University Press, England, 2nd edition, 2009.

[5] Spivey JM. Understanding Z: A Specification Language and its Formal Semantics, volume 3 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, January 1988.

[6] Spivey JM. The Z Notation: A Reference Manual. Prentice Hall, Hemel Hampstead, England. International Series in Computer Science, 2nd edition, 1992.

[7] Woodcock JCP, Davies J. Using Z: Specification, Proof and Refinement. Prentice Hall, Hemel Hampstead, England. International Series in Computer Science, 1996.

[8] Abrial JR. The B Book: Assigning Programs to Meanings. Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, England, 1996.

[9] Abrial JR. Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge, England, 2009.

[10] George CW, Haff PL, Havelund K, Haxthausen AE, Milne R, Nielsen CB, Prehn S, Wagner KR. The RAISE Specification Language. The BCS Practitioner Series. Prentice Hall, Hemel Hampstead, England, 1992.

[11] George CW, Haxthausen AE, Hughes S, Milne R, Prehn S, Pedersen JS. The RAISE Method. The BCS Practitioner Series. Prentice Hall, Hemel Hampstead, England, 1995.

[12] Bjørner D. Software Engineering, Vol. 1: Abstraction and Modelling. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.
[13] Bjørner D. Software Engineering, Vol. 2: Specification of Systems and Languages. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. Chapters 12–14 are primarily authored by Christian Krog Madsen.
[14] Bjørner D. Software Engineering, Vol. 3: Domains, Requirements and Software Design. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.
[15] Bjørner D. Summer 2009c, From Domains to Requirements: The Triptych Approach to Software Engineering. Submitted to Springer for evaluation in 2009. Slightly incomplete draft version, approximately XXVII+160+25 pages (frontmatter, main text, appendices). See http://www.imm.dtu/~db/de+re-p.pdf.
[16] van Eijk PHJ , Vissers CA, Diaz M. eds. The formal description technique LOTOS. Elsevier Science Publishers, Amsterdam, 1989.
[17] Zhou CC, Hoare CAR , Ravn AP. A Calculus of Durations. Information Proc. Letters, 1992, 40(5).
[18] Zhou CC, Hansen MR. Duration Calculus: A Formal Approach to Real–time Systems. Monographs in Theoretical Computer Science, the EATCS Series. Springer, 2004.
[19] Reisig W. Logics of Specification Languages, chapter Abstract State Machines for the Classroom. Springer, 2008.15–46.
[20] Jackson D. Software Abstractions Logic, Language, and Analysis. The MIT Press, Cambridge, Mass., USA, April 2006.
[21] Futatsugi K, Nakagawa AT, Tamai T. eds. CAFE: An Industrial–Strength Algebraic Formal Method, Amsterdam, The Netherlands, 2000. Elsevier. Proceedings from an April 1998 Symposium, Numazu, Japan.
[22] Futatsugi K, Diaconescu R. CafeOBJ Report The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification. AMAST Series in Computing, World Scientific Publishing Co. 1998. 6.
[23] Bidoit M, Mosses PD. CASL User Manual. Lecture Notes in Computer Science, vol. 2900 (IFIP Series). Springer, 2004.
[24] CoFI (The Common Framework Initiative). CASL Reference Manual, Lecture Notes in Computer Science, the IFIP Series, Springer, 2004. 2960.
[25] Hoare CAR. Communicating Sequential Processes. Hoare CAR Series in Computer Science. Prentice Hall International, Hemel Hampstead, England, 1985.
[26] Roscoe AW. Theory and Practice of Concurrency. Hoare CAR Series in Computer Science. Prentice Hall, 1997. Now available on the net: http://www.comlab.ox.ac.uk/people/bill.roscoe/publications/68b.pdf.
[27] Schneider S. Concurrent and Real-time Systems – The CSP Approach. Worldwide Series in Computer Science. John Wiley & Sons, Ltd., Chichester, England, January 2000.
[28] Hoare CAR. Communicating Sequential Processes. Published electronically: http://www.-usingcsp.com/cspbook.pdf, 2004. Second edition of [25]. See also http://www.usingcsp.com/.
[29] Harel D, Marelly R. Come, Let's Play – Scenario-Based Programming Using LSCs and the Play-Engine. Springer, 2003.
[30] ITU-T, CCITT Recommendation Z.120: Message Sequence Chart (MSC), 1992.
[31] ITU-T. ITU-T Recommendation Z.120: Message Sequence Chart (MSC), 1996.
[32] ITU-T. ITU-T Recommendation Z.120: Message Sequence Chart (MSC), 1999.
[33] W. Reisig. A Primer in Petri Net Design. Springer, March 1992. 120 pages.
[34] Reisig W. Petri Nets: An Introduction, volume 4 of EATCS Monographs in Theoretical Computer Science. Springer, May 1985.
[35] Reisig W. Elements of Distributed Algorithms: Modelling and Analysis with Petri Nets. Springer, December 1998. xi + 302 pages.
[36] Harel D. Statecharts: A visual formalism for complex systems. Science of Computer Programming, 1987, 8(3): 231–274.
[37] Manna Z, Pnueli A. The Temporal Logic of Reactive Systems: Specifications. Addiso–Wesley, 1991.

[38] Manna Z, Pnueli A. The Temporal Logic of Reactive Systems: Safety. Addison–Wesley, 1995.
[39] Moszkowski BC. Executing Temporal Logic Programs. Cambridge University Press, Cambridge, England, 1986.
[40] Pnueli A. The Temporal Logic of Programs. In Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, IEEE CS FoCS. Providence, Rhode Island, IEEE CS, 1977. 46–57.
[41] Bjørner NS, Browne A, Colon M, Finkbeiner B, Manna Z, Sipma H, Uribe T. Verifying Temporal Properties of Reactive Systems: A STeP Tutorial. Formal Methods in System Design, 2000, 16: 227–270.
[42] Manna Z, Anuchitanukul A, Bjørner NS, Browne A, Chang E, Colon MI, de Alfaro L, Devarajan H, Sipma H, Uribe T. STeP: The Stanford Temporal Prover. Technical Report No. STAN-CS-TR-94-1518, Computer Science Department, Stanford University, California, USA, July 1994.
[43] Lamport L. The Temporal Logic of Actions. Transactions on Programming Languages and Systems, 1995, 16(3): 872–923.
[44] Lamport L. Specifying Systems. Addison–Wesley, Boston, Mass., USA, 2002.
[45] Bjørner D, Henson MC. eds. Logics of Specification Languages. Monograph in Theoretical Computer Science, the EATCS Series. Springer, Heidelberg, Germany, 2008.
[46] Bjørner D. On Mereologies in Computing Science. In Festschrift for Tony Hoare, History of Computing (ed. Roscoe B), Springer, (expected) 2009.
[47] Bjørner D. An Emerging Domain Science – A Rôle for Stanisław Leśniewski's Mereology and Bertrand Russell's Philosophy of Logical Atomism, Higher-order and Symbolic Computation. Springer, 2009, 22(3–4).

Dines Bjørner received his MSc and PhD degrees from the Technical University of Denmark in 1962, respectively in 1969. From 1962 to 1975 he was with IBM at their Nordic Labs. (Stockholm, Sweden, 1962–63), Syst. Devt. Lab. (San Jose, Calif., USA. 1963–65), European Systems Research Inst. (Geneva, Switzerland, 1967–68), Adv. Sys. Lab. (Menlo Park, Calif., USA, 1969), Research Lab. (San Jose, Calif., USA, 1969-73), and the Vienna Laboratory (Austria, 1973–75). Bjørner was a visiting prof. at UC Berkeley 1971–72; 1975-76 he was a visiting prof. at Copenhagen Univ., Denmark – before taking up a chair at The Technical University of Denmark, Kgs. Lyngby (1976–2007). In 1980 Bjørner was a visiting prof. at Carl Albrechts Univ. (Kiel, Germany). In 1979 Bjørner was a co-founder of Dansk Datamatik Center (DDC) and was its chief scientist during the 10 year life of DDC. DDC became known for its work on the Formal Definition of Ada, on the first European Ada compiler, and on the research and development which resulted in RAISE (rigorous approach to industrial software engineering)[10,11]. From 1991 to 1997 Bjørner was the founding and first UN Director of UNU-IIST, the United Nations International Institute for Software Technology, Macau. 2004–05 Bjørner was a visiting prof. at NUS: National University of Singapore; and in 2006 he was for 12 month a visiting prof. at JAIST, Japan Adv. Inst. of Sci. and Techn., Kanazawa. After his retirement, in his 71st–72nd year, Bjørner has lectured[15] at Henri Poincare Univ. and at INRIA, Nancy, France; Techn. Univ. of Graz, Austria; and at Univ. of Saarland, Saarbrücken, Germany. At the moment he is distinguished visitor at the Univ. of Edinburgh, Scotland and will later, in late 2009, spend a month at Tokyo Univ., Japan. Bjørner's research, since 1973, has been focused on formal techniques for software development. He is a co-instigator of VDM and RAISE, has published some 130 papers, edited and co-authored some 12 books and written a three volume textbook cum monograph on software engineering[12–14]. A new, approx. 200 page book is currently being evaluated for publication[15]. Recently Bjørner's research has moved into a border-line field of computing science and an emerging Philosophy of Informatics[46,47]. Over the years Bjørner has served and serves on the editorial board of more than 10 international science journals, and co-founded Formal Methods Europe (in 1987 with Cliff Jones, as VDM Europe). Bjørner is a Fellow of ACM and of IEEE, a member of the Danish Academy of Technical Sciences (ATV), Academia Europaea (AE), and of the Russian Academy of Natural Sciences (AB). In 1993 he received the John von Neumann medal from the Hungarian John von Neumann society, in 1996 the Masaryk Gold Medal from Masaryk Univ., Brno, The Czech Republic, and is a Dr.h.c. of that university (2003). In 1997 Bjørner received the BIT prize of the Danish Society of Engineers.