5 Conclusion 226

We discuss a number of issues.

5.1 What Have We Omitted

Our coverage of domain and requirements engineering has focused on modelling techniques for domain and requirements facets. We have omitted the important software engineering tasks of **stakeholder identification and liaison**, **domain** and, to some extents also **requirements**, especially **goal acquisition and analysis**, **terminologisation**, and techniques for **domain and requirements and goal validation and [goal] verification** ($\mathcal{D}, \mathcal{R} \models \mathcal{G}$).

We refer, instead, to [32, Vol.3, Part IV (Chaps. 9, 12–14) and Part V (Chaps. 18, 20–23)].

5.2 Domain Descriptions Are Not Normative 227

A description of, for example, “the” domain of the *New York Stock Exchange* would describe the set of rules and regulations governing the submission of sell offers and buy bids as well as rules and regulations for clearing (‘matching’) sell offers and buy bids. These rules and regulations appears to be quite different from those of the *Tokyo Stock Exchange* [218]. A normative description of stock exchanges would abstract these rules so as to be rather un-informative. And, anyway, rules and regulations changes and business process re-engineering changes entities, actions, events and behaviours. For any given software development one may thus have to rewrite parts of existing domain descriptions, or construct an entirely new such description.

5.3 “Requirements Always Change” 228

This claim is often used as a hidden excuse for not doing a proper, professional job of requirements prescription, let alone “deriving” them, as we advocate, from domain descriptions. Instead we now make the following counterclaims [1] “domains are far more stable than requirements” and [2] “requirements changes arise more as a result of business process re-engineering than as a result of changing stakeholder ideas”. 229

Closer studies of a number of domain descriptions, for example of a *financial service industry*, reveals that the domain in terms of which an “ever expanding” variety of financial products are offered, are, in effect, based on a small set of very basic domain functions which have been offered for well-nigh centuries !

We thus claim that thoroughly developed domain descriptions and thoroughly “derived” requirements prescriptions tend to stabilise the requirements re-design, but never alleviate it.

5.4 What Can Be Described and Prescribed 230

The issue of *"what can be described"* has been a constant challenge to philosophers. In [205, 1919] Bertrand Russell covers his first *Theory of Descriptions*, and in [204, Philosophy of Mathematics] a revision, as *The Philosophy of Logical Atomism*. The issue is not that straightforward. In [40, 41] we try to broach the topic from the point of view of the kind of domain engineering presented in this paper.

Our approach is simple; perhaps too simple ! We can describe what can be observed. We do so, first by postulating types of observable phenomena and of derived concepts; then 231
by the introduction of *observer* functions and by axioms over these, that is, over values of postulated types and observers. To this we add defined functions; usually described by pre/post-conditions. The narratives refer to the "real" phenomena whereas the formalisations refer to related phenomenological concepts. The narrative/formalisation problem is that one can 'describe' phenomena without always knowing how to formalise them.

5.5 What Have We Achieved – and What Not 232

Section 1.2.3 made some claims. We think we have substantiated them all, albeit ever so briefly.

Each of the domain facets (intrinsics, support technologies, rules and regulations, scripts [licenses and contracts], management and organisation and human behaviour) and each of the requirements facets (projection, instantiation, determination, extension and fitting) provide rich grounds for both specification methodology studies and and for more theoretical studies [35, ICTAC 2007].

5.6 Relation to Other Work 233

The most obvious 'other' work is that of [140, Problem Frames]. In [140] Jackson, like is done here, departs radically from conventional requirements engineering. In his approach understandings of the domain, the requirements and possible software designs are arrived at, not hierarchically, but in parallel, interacting streams of decomposition. Thus 234
the 'Problem Frame' development approach iterates between concerns of domains, requirements and software design. "Ideally" our approach pursues domain engineering prior to requirements engineering, and, the latter, prior to software design. But see next. 235

The recent book [149, Axel van Lamsweerde] appears to represent the most definitive work on Requirements Engineering today. Much of its requirements and goal acquisition and analysis techniques carries over to main aspects of domain acquisition and analysis techniques and the goal-related techniques of [149] apply to determining which projections, instantiation, determination and extension operations to perform on domain descriptions.

5.7 "Ideal" Versus Real Developments 236

The term 'ideal' has been used in connection with 'ideal development' from domain to requirements. We now discuss that usage. Ideally software development could proceed from developing domain descriptions via "deriving" requirements prescriptions to software design, each phase involving extensive formal specifications, verifications (formal testing, model checking and theorem proving) and validation. 237

More realistically less comprehensive domain description development (D) may alternate with both requirements development (R) work and with software design (S) – in some controlled, contained iterated and "spiralling" manner and such that it is at all times clear which development step is what: $\mathcal{D}$, $\mathcal{R}$ or $\mathcal{S}$!

5.8 Description Languages 238

We have used the `RSL` specification language, [110, 32], for the formalisations of this report, but any of the model-oriented approaches and languages offered by `Alloy` [138], `B, Event B` [3], `RAISE` [112], `VDM` [107] and `Z` [234], should work as well. 239

No single one of the above-mentioned formal specification languages, however, suffices. Often one has to carefully combine the above with elements of `Petri Nets` [200], `CSP` [128], `MSC` [137], `Statecharts` [120], and/or some temporal logic, for example either `DC` [236] or `TLA+` [148]. Research into how such diverse textual and diagrammatic languages can be combined is ongoing [9].

5.9 Entailments 240

$\mathcal{D},\mathcal{R} \models \mathcal{G}$[*] From the $\mathcal{D}$omain and the $\mathcal{R}$equirements we can reason that the $\mathcal{G}$oals are met.

$\mathcal{D},\mathcal{S} \models \mathcal{R}$[*] In a proof of correctness of $\mathcal{S}$oftware design with respect to $\mathcal{R}$equirements prescriptions one often has to refer to assumptions about the $\mathcal{D}$omain. [*] Formalising our understandings of the $\mathcal{D}$omain, the $\mathcal{R}$equirements and the $\mathcal{S}$oftware design enables proofs that the software is right and the formalisation of the "derivation" of $\mathcal{R}$equirements from $\mathcal{D}$omain specifications help ensure that it is the right software [58].

5.10 Domain Versus Ontology Engineering 241

In the information science community an ontology is a "formal, explicit specification of a shared conceptualisation". Most of the information science ontology work seems aimed primarily at axiomatisations of properties of entities. Apart from that there are many issues of "ontological engineering" that are similar to the triptych kind of domain engineering; but then, we claim, that domain engineering goes well beyond ontological engineering and makes free use of whatever formal specification languages are needed, cf. Sect. 6.1.

6 Bibliographical Notes 242

6.1 Description Languages

Besides using as precise a subset of a national language, as here English, as possible, and in enumerated expressions and statements, we have "paired" such narrative elements with corresponding enumerated clauses of a formal specification language. We have been using the `RAISE` Specification Language, `RSL`, [112], in our formal texts. But any of the model-oriented approaches and languages offered by

- `Alloy` [138],
- `CafeOBJ` [109],
- `Event B` [3],
- `VDM` [107] and
- `Z` [234],

should work as well. 243

No single one of the above-mentioned formal specification languages, however, suffices. Often one has to carefully combine the above with elements of

- `Petri Nets` [200],
- `CSP: Communicating Sequential Processes` [128],
- `MSC: Message Sequence Charts` [137],
- `Statecharts` [120],
- and some temporal logic, for example
  - `DC: Duration Calculus` [236]
  - or `TLA+` [148].

Research into how such diverse textual and diagrammatic languages can be meaningfully and proof-theoretically combined is ongoing [9]. And even then !

6.2 References

[1] H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. The MIT Press, Cambridge, Mass., USA, 1996. 2nd edition.

[2] J.-R. Abrial. *The B Book: Assigning Programs to Meanings*. Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, England, 1996.

[3] J.-R. Abrial. The B Book: Assigning Programs to Meanings *and* Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge, England, 1996 and 2009.

[4] J.-R. Abrial and L. Mussat. *Event B Reference Manual (Editor: Thierry Lecomte)*, June 2001. Report of EU IST Project Matisse IST-1999-11435.

[5] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994. (Preliminary versions appeared in Proc. 17th ICALP, LNCS 443, 1990, and Real Time: Theory in Practice, LNCS 600, 1991).

[6] J. Alves-Foss, editor. *Formal Syntax and Semantics of Java*. LNCS. Springer–Verlag, 1998.

[7] D. Andrews and W. Henhapl. Pascal. In *[53]*, chapter 7, pages 175–252. Prentice-Hall, 1982.

[8] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, August 2003. ISBN 0521825830.

[9] K. Araki et al., editors. *IFM 1999–2009: Integrated Formal Methods*, volume 1945, 2335, 2999, 3771, 4591, 5423 (only some are listed) of *Lecture Notes in Computer Science*. Springer, 1999–2009.

[10] K. Arnold, J. Gosling, and D. Holmes. *The Java Programming Language*. Addison Wesley, US, 1996.

[11] R.-J. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. Springer-Verlag, Heidelberg, Germany, 1998.

[12] J. W. Backus and P. Naur. Revised Report on the Algorithmic Language ALGOL 60. *Communications of the ACM*, 6(1):1–1, 1963.

[13] H. P. Barendregt. *The Lambda Caculus — Its Syntax and Semantics*. North-Holland Publ.Co., Amsterdam, 1981.

[14] H. P. Barendregt. Introduction to Lambda Calculus. *Niew Archief Voor Wiskunde*, 4:337–372, 1984.

[15] H. P. Barendregt. *The Lambda Calculus*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, revised edition, 1991.

[16] H. Barringer, J. Cheng, and C. B. Jones. A logic covering undefinedness in program proofs. *Acta Informatica*, 21:251–269, 1984.

[17] H. Bekič, P. Lucas, K. Walk, and M. Others. Formal Definition of PL/I, ULD Version I. Technical report, IBM Laboratory, Vienna, 1966.

[18] H. Bekič, P. Lucas, K. Walk, and M. Others. Formal Definition of PL/I, ULD Version II. Technical report, IBM Laboratory, Vienna, 1968.

[19] H. Bekič, P. Lucas, K. Walk, and M. Others. Formal Definition of PL/I, ULD Version III. IBM Laboratory, Vienna, 1969.

[20] C. Berge. *Théorie des Graphes et ses Applications*. Collection Universitaire de Mathematiques. Dunod, Paris, 1958. See [21].

[21] C. Berge. *Graphs*, volume 6 of *Mathematical Library*. North-Holland Publ. Co., second revised edition of part 1 of the 1973 english version edition, 1985. See [20].

[22] R. Bird and O. de Moor. *Algebra of Programming*. Prentice Hall, September 1996.

[23] G. Birtwistle, O.-J.Dahl, B. Myhrhaug, and K. Nygaard. *SIMULA* `begin`. Studentlitteratur, Lund, Sweden, 1974.

[24] D. Bjørner. Programming in the Meta-Language: A Tutorial. In D. Bjørner and C. B. Jones, editors, *The Vienna Development Method: The Meta-Language, [52]*, LNCS, pages 24–217. Springer–Verlag, 1978.

[25] D. Bjørner. Software Abstraction Principles: Tutorial Examples of an Operating System Command Language Specification and a PL/I-like On-Condition Language Definition. In D. Bjørner and C. B. Jones, editors, *The Vienna Development Method: The Meta-Language, [52]*, LNCS, pages 337–374. Springer–Verlag, 1978.

[26] D. Bjørner. The Vienna Development Method: Software Abstraction and Program Synthesis. In *Mathematical Studies of Information Processing*, volume 75 of *LNCS*. Springer–Verlag, 1979. Proceedings of Conference at Research Institute for Mathematical Sciences (RIMS), University of Kyoto, August 1978.

[27] D. Bjørner, editor. *Abstract Software Specifications*, volume 86 of *LNCS*. Springer, 1980.

[28] D. Bjørner. Application of Formal Models. In *Data Bases*. INFOTECH Proceedings, October 1980.

[29] D. Bjørner. Formalization of Data Base Models. In D. Bjørner, editor, *Abstract Software Specification, [27]*, volume 86 of *LNCS*, pages 144–215. Springer–Verlag, 1980.

[30] D. Bjørner. Domain Modelling: Resource Management Strategics, Tactics & Operations, Decision Support and Algorithmic Software. In J. Davies, B. Roscoe, and J. Woodcock, editors, *Millenial Perspectives in Computer Science*, Cornerstones of Computing (Ed.: Richard Bird and Tony Hoare), pages 23–40, Houndmills, Basingstoke, Hampshire, RG21 6XS, UK, 2000. Palgrave (St. Martin's Press). An Oxford University and Microsoft Symposium in Honour of Sir Anthony Hoare, September 13–14, 1999.

[31] D. Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. See [37, 42].

[32] D. Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling; Vol. 2: Specification of Systems and Languages; ol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

[33] D. Bjørner. *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. See [38, 43].

[34] D. Bjørner. *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. See [39, 44].

[35] D. Bjørner. Domain Theory: Practice and Theories, Discussion of Possible Research Topics. In *ICTAC'2007*, volume 4701 of *Lecture Notes in Computer Science (eds. J.C.P. Woodcock et al.)*, pages 1–17, Heidelberg, September 2007. Springer.

[36] D. Bjørner. From Domains to Requirements. In *Montanari Festschrift*, volume 5065 of *Lecture Notes in Computer Science (eds. Pierpaolo Degano, Rocco De Nicola and José Meseguer)*, pages 1–30, Heidelberg, May 2008. Springer.

[37] D. Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press, 2008.

[38] D. Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press, 2008.

[39] D. Bjørner. *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press, 2008.

[40] D. Bjørner. An Emerging Domain Science – A Rôle for Stanisław Leśniewski's Mereology and Bertrand Russell's Philosophy of Logical Atomism. *Higher-order and Symbolic Computation*, 2009.

[41] D. Bjørner. On Mereologies in Computing Science. In *Festschrift for Tony Hoare*, History of Computing (ed. Bill Roscoe), London, UK, 2009. Springer.

[42] D. Bjørner. **Chinese:** *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[43] D. Bjørner. **Chinese:** *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[44] D. Bjørner. **Chinese:** *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[45] D. Bjørner. Domain Engineering. In *BCS FACS Seminars*, Lecture Notes in Computer Science, the BCS FAC Series (eds. Paul Boca and Jonathan Bowen), pages 1–42, London, UK, 2010. Springer.

[46] D. Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics, Part I of II: The Engineering Part*. *Kibernetika i sistemny analiz*, (2), May 2010.

[47] D. Bjørner. Believable Software Management. *Encyclopedia of Software Engineering*, 1(1):1–32, 2011.

[48] D. Bjørner. *Domain Engineering: Technology Management, Research and Engineering*. A JAIST Press Research Monograph (#4), March 2009. This Research Monograph[12] contains the following chapters: [83, 85, 86, 87, 88, 89, 90, 91, 92, 84]. Bjørner will post this 507 page book (with 77 fine photos of "all things Japanese", in full colours, taken by Dines in 2006) to you provided you e-mail your name and address and post international reply postage coupons (http://en.wikipedia.org/wiki/International_reply_coupon) to DINES BJØRNER, FREDSVEJ 11, DK-2840 HOLTE, DENMARK in the total amount of: Denmark 60.50 Kr., Europe 126.00 Kr., elsewhere 209.00 Kr.

[49] D. Bjørner. From Domains to Requirements — On a Triptych of Software Development. *Submitted for publication*, Submitted 8 January, 2010.

[50] D. Bjørner. The Role of Domain Engineering in Software Development. Why Current Requirements Engineering Seems Flawed! In *Perspectives of Systems Informatics*, volume 5947 of *Lecture Notes in Computer Science*, pages 2–34, Heidelberg, Wednesday, January 27, 2010. Springer.

[51] D. Bjørner, A. P. Ershov, and N. D. Jones, editors. *Partial Evaluation and Mixed Computation. Proceedings of the IFIP TC2 Workshop, Gammel Avernæs, Denmark, October 1987*. North-Holland, 1988. 625 pages.

[52] D. Bjørner and C. B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *LNCS*. Springer, 1978. This was the first monograph on *Meta-IV*. [24, 25, 26].

[53] D. Bjørner and C. B. Jones, editors. *Formal Specification and Software Development*. Prentice-Hall, 1982.

[54] D. Bjørner and H. H. Løvengreen. Formal Semantics of Data Bases. In *8th Int'l. Very Large Data Base Conf.*, Mexico City, Sept. 8-10 1982.

[55] D. Bjørner and H. H. Løvengreen. Formalization of Data Models. In *Formal Specification and Software Development, [53]*, chapter 12, pages 379–442. Prentice-Hall, 1982.

[12] http://www.imm.dtu.dk/ db/jaistmono.pdf

[56] D. Bjørner and O. N. Oest, editors. *Towards a Formal Description of Ada*, volume 98 of *LNCS*. Springer, 1980.

[57] W. D. Blizard. A Formal Theory of Objects, Space and Time. *The Journal of Symbolic Logic*, 55(1):74–89, March 1990.

[58] B. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ., USA, 1981.

[59] R. Bruni and J. Meseguer. Generalized Rewrite Theories. In Jos C. M. Baeten and Jan Karel Lenstra and Joachim Parrow and Gerhard J. Woeginger, editor, *Automata, Languages and Programming. 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 2003.

[60] D. Cansell and D. Méry. Logical Foundations of the B Method. *Computing and Informatics*, 22(1–2), 2003.

[61] D. Carrington, D. J. Duke, R. Duke, P. King, G. A. Rose, and G. Smith. Object-Z: An object-oriented extension to Z. In S. Vuong, editor, *Formal Description Techniques, II (FORTE'89)*, pages 281–296. Elsevier Science Publishers (North-Holland), 1990.

[62] C.C.I.T.T. The Specification of CHILL. Technical Report Recommendation Z200, International Telegraph and Telephone Consultative Committee, Geneva, Switzerland, 1980.

[63] E. Chailloux, P. Manoury, and B. Pagano. *Developing Applications With Objective Caml*. Project Cristal, INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, F-78153 Le Chesnay Cedex, France, 2004. Preliminary translation of the book Développement d'applications avec Objective Caml [64].

[64] E. Chailloux, P. Manoury, and B. Pagano. *Développement d'applications avec Objective Caml*. Éditions O'Reilly, Paris, France, Avril 2000. ISBN 2-84177-121-0.

[65] J. Cheng. *A Logic for Partial Functions*. PhD thesis, Department of Computer Science, University of Manchester, 1986. UMCS-86-7-1.

[66] A. Church. *Introduction to Mathematical Logic*. The Princeton University Press, Princeton, New Jersey, USA, 1956. Reprint Edition, October 28, 1996, ISBN 0691029067.

[67] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 System. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in Lecture Notes in Computer Science, pages 76–87. Springer-Verlag, June 2003.

[68] E. F. Codd. A relational model for large shared databank. *Communications of the ACM*, 13(6):377–387, 1970.

[69] G. Cousineau and M. Mauny. *The Functional Approach to Programming*. Cambridge University Press, Cambridge, UK, 1998. ISBN 0-521-57183-9 (hardcover), 0-521-57681-4 (paperback).

[70] D. Crystal. *The Cambridge Encyclopedia of Language*. Cambridge University Press, 1987, 1988.

[71] O.-J. Dahl, E. Dijkstra, and C. Hoare. *Structured Programming*. Academic Press, 1972.

[72] O.-J. Dahl and C. Hoare. Hierarchical program structures. In *[71]*, pages 197–220. Academic Press, 1972.

[73] O.-J. Dahl and K. Nygaard. SIMULA – an ALGOL-based simulation language. *Communications of the ACM*, 9(9):671–678, 1966.

[74] O. Danvy. A Rational Deconstruction of Landin's SECD Machine. Research RS 03–33, BRICS: Basic Research in Computer Science, Dept. of Comp.Sci., University of Århus, Ny Munkegade, Bldg. 540, DK-8000 Århus C, Denmark, October 2003. ISSN 0909 0878. E–mail: BRICS@brics.dk.

[75] C. Date. *An Introduction to Database Systems, I*. The Systems Programming Series. Addison Wesley, 1981.

[76] C. Date. *An Introduction to Database Systems, II*. The Systems Programming Series. Addison Wesley, 1983.

[77] C. Date and H. Darwen. *A Guide to the SQL Standard*. Addison-Wesley Professional, November 8, 1996. 4th Edition, ISBN: 0201964260.

[78] J. de Bakker. *Control Flow Semantics*. The MIT Press, Cambridge, Mass., USA, 1995.

[79] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier, 1990.

[80] R. Diaconescu and K. Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*. AMAST Series in Computing - Vol. 6. World Scientific Publishing Co., Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, July 1998. 196pp, ISBN 981-02-3513-5, US$30.

[81] R. Diaconescu, K. Futatsugi, and K. Ogata. CafeOBJ: Logical Foundations and Methodology. *Computing and Informatics*, 22(1–2), 2003.

[82] E. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[83] Dines Bjørner. *[48] Chap. 1: On Domains and On Domain Engineering – Prerequisites for Trustworthy Software – A Necessity for Believable Management*, pages 3–38. JAIST Press, March 2009.

[84] Dines Bjørner. *[48] Chap. 10: Towards a Family of Script Languages – – Licenses and Contracts – Incomplete Sketch*, pages 283–328. JAIST Press, March 2009.

[85] Dines Bjørner. *[48] Chap. 2: Possible Collaborative Domain Projects – A Management Brief*, pages 39–56. JAIST Press, March 2009.

[86] Dines Bjørner. *[48] Chap. 3: The Rôle of Domain Engineering in Software Development*, pages 57–72. JAIST Press, March 2009.

[87] Dines Bjørner. *[48] Chap. 4: Verified Software for Ubiquitous Computing – A VSTTE Ubiquitous Computing Project Proposal*, pages 73–106. JAIST Press, March 2009.

[88] Dines Bjørner. *[48] Chap. 5: The Triptych Process Model – Process Assessment and Improvement*, pages 107–138. JAIST Press, March 2009.

[89] Dines Bjørner. *[48] Chap. 6: Domains and Problem Frames – The Triptych Dogma and M.A.Jackson's PF Paradigm*, pages 139–175. JAIST Press, March 2009.

[90] Dines Bjørner. *[48] Chap. 7: Documents – A Rough Sketch Domain Analysis*, pages 179–200. JAIST Press, March 2009.

[91] Dines Bjørner. *[48] Chap. 8: Public Government – A Rough Sketch Domain Analysis*, pages 201–222. JAIST Press, March 2009.

[92] Dines Bjørner. *[48] Chap. 9: Towards a Model of IT Security — – The ISO Information Security Code of Practice – An Incomplete Rough Sketch Analysis*, pages 223–282. JAIST Press, March 2009.

[93] O. Dommergaard. The design of a virtual machine for Ada. In *[27]*, pages 463–605. Springer, 1980.

[94] O. Dommergaard and S. Bodilsen. A formal definition of P-code. Technical report, Dept. of Comp. Sci., Techn. Univ. of Denmark, 1980.

[95] D. J. Duke and R. Duke. Towards a semantics for Object-Z. In D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors, *VDM and Z – Formal Methods in Software Development*, volume 428 of *Lecture Notes in Computer Science*, pages 244–261. VDM-Europe, Springer-Verlag, 1990.

[96] R. Duke, P. King, G. A. Rose, and G. Smith. The Object-Z specification language. In T. Korson, V. Vaishnavi, and M. B, editors, *Technology of Object-Oriented Languages and Systems: TOOLS 5*, pages 465–483. Prentice Hall, 1991.

[97] R. K. Dybvig. *The Scheme Programming Language*. The MIT Press, Cambridge, Mass., USA, 2003. 3rd Edition.

[98] A. Ershov. On the essence of translation. *Computer Software and System Programming*, 3(5):332–346, 1977.

[99] A. Ershov. On the partial computation principle. *Information Processing Letters*, 6(2):38–41, April 1977.

[100] A. Ershov. Mixed computation: Potential applications and problems for study. *Theoretical Computer Science*, 18:41–67, 1982.

[101] A. Ershov. On Futamura projections. *BIT (Japan)*, 12(14):4–5, 1982. (In Japanese).

[102] A. Ershov. On mixed computation: Informal account of the strict and polyvariant computational schemes. In M. Broy, editor, *Control Flow and Data Flow: Concepts of Distributed Programming. NATO ASI Series F: Computer and System Sciences, vol. 14*, pages 107–120. Springer-Verlag, 1985.

[103] A. Ershov, D. Bjørner, Y. Futamura, K. Furukawa, A. Haraldson, and W. Scherlis, editors. *Special Issue: Selected Papers from the Workshop on Partial Evaluation and Mixed Computation, 1987 (New Generation Computing, vol. 6, nos. 2,3)*. Ohmsha Ltd. and Springer-Verlag, 1988.

[104] A. Ershov and V. Grushetsky. An implementation-oriented method for describing algorithmic languages. In B. Gilchrist, editor, *Information Processing 77, Toronto, Canada*, pages 117–122. North-Holland, 1977.

[105] A. Ershov and V. Itkin. Correctness of mixed computation in Algol-like programs. In J. Gruska, editor, *Mathematical Foundations of Computer Science, Tatranská Lomnica, Czechoslovakia. (Lecture Notes in Computer Science, vol. 53)*, pages 59–77. Springer-Verlag, 1977.

[106] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142, 1996. 2nd printing.

[107] J. Fitzgerald and P. G. Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, Cambridge, UK, Second edition, 2009.

[108] FOLDOC: The free online dictionary of computing. Electronically, on the Web: `http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?ISWIM`, 2004.

[109] K. Futatsugi, A. Nakagawa, and T. Tamai, editors. *CAFE: An Industrial–Strength Algebraic Formal Method*, Sara Burgerhartstraat 25, P.O. Box 211, NL–1000 AE Amsterdam, The Netherlands, 2000. Elsevier. Proceedings from an April 1998 Symposium, Numazu, Japan.

[110] C. W. George, P. Haff, K. Havelund, A. E. Haxthausen, R. Milne, C. B. Nielsen, S. Prehn, and K. R. Wagner. *The RAISE Specification Language*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.

[111] C. W. George and A. E. Haxthausen. The Logic of the RAISE Specification Language. *Computing and Informatics*, 22(1–2), 2003.

[112] C. W. George, A. E. Haxthausen, S. Hughes, R. Milne, S. Prehn, and J. S. Pedersen. *The RAISE Development Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.

[113] J. Gosling and F. Yellin. *The Java Language Specification*. Addison-Wesley & Sun Microsystems. ACM Press Books, 1996. 864 pp, ISBN 0-10-63451-1.

[114] D. Gries. *The Science of Programming*. Springer-Verlag, 1981.

[115] O. Grillmeyer. *Exploring Computer Science with Scheme*. Springer-Verlag, New York, USA, 1998.

[116] C. Gunther. *Semantics of Programming Languages*. The MIT Press, Cambridge, Mass., USA, 1992.

[117] Y. Gurevich. Sequential Abstract State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic*, 1(1):77–111, July 2000.

[118] P. Haff, editor. *The Formal Definition of CHILL*. ITU (Intl. Telecmm. Union), Geneva, Switzerland, 1981.

[119] M. R. Hansen and H. Rischel. *Functional Programming in Standard ML*. Addison Wesley, 1997.

[120] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[121] D. Harel and R. Marelly. *Come, Let's Play – Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag, 2003.

[122] F. Harrary. *Graph Theory*. Addison Wesley Publishing Co., 1972.

[123] E. Hehner. *The Logic of Programming*. Prentice-Hall, 1984.

[124] E. Hehner. *a Practical Theory of Programming*. Springer-Verlag, 2nd edition, 1993. On the net: http://www.cs.toronto.edu/~hehner/aPToP/.

[125] A. Hejlsberg, S. Wiltamuth, and P. Golde. *The C# Programming Language*. Microsoft •net Development Series. Addison-Wesley, 75 Arlington Street, Suite 300, Boston, MA 02116, USA, (617) 848-6000, 30 October 2003. 672 page, ISBN 0321154916.

[126] M. C. Henson, S. Reeves, and J. P. Bowen. Z Logic and its Consequences. *Computing and Informatics*, 22(1–2), 2003.

[127] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, N.Y., USA, June 1962. ASIN 0801401879.

[128] C. Hoare. *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985. Published electronically: http://www.usingcsp.-com/cspbook.pdf (2004).

[129] C. Hoare and N. Wirth. An axiomatic definition of the programming language Pascal. *Acta Informatica*, 2:335–355, 1973.

[130] T. Hoare. *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985.

[131] T. Hoare. Communicating Sequential Processes. Published electronically: `http://www.-usingcsp.com/cspbook.pdf`, 2004. Second edition of [130]. See also `http://www.-usingcsp.com/`.

[132] D. M. Hoffman and D. M. Weiss, editors. *Software fundamentals: collected papers by David L. Parnas*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[133] C. J. Hogger. *Essentials of Logic Programming*. Graduate Texts in Computer Science, no.1, 310 pages. Clarendon Press, December 1990. .

[134] IEEE CS. IEEE Standard Glossay of Software Engineering Terminology, 1990. IEEE Std.610.12.

[135] Inmos Ltd. Specification of instruction set & Specification of floating point unit instructions. In *Transputer Instruction Set – A compiler writer's guide*, pages 127–161. Prentice Hall, Hemel Hempstead, Hertfordshire HP2 4RG, UK, 1988.

[136] B. B. S. Institution. Specification for computer programming language Pascal. Technical Report BS6192, BSI, 1982.

[137] ITU-T. CCITT Recommendation Z.120: Message Sequence Chart (MSC), 1992, 1996, 1999.

[138] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, Cambridge, Mass., USA, April 2006. ISBN 0-262-10114-9.

[139] M. A. Jackson. *Software Requirements & Specifications: a lexicon of practice, principles and prejudices*. ACM Press. Addison-Wesley, Reading, England, 1995.

[140] M. A. Jackson. *Problem Frames — Analyzing and Structuring Software Development Problems*. ACM Press, Pearson Education. Addison-Wesley, England, 2001.

[141] K. Jensen and N. Wirth. *Pascal User Manual and Report*, volume 18 of *LNCS*. Springer–Verlag, 1976.

[142] N. D. Jones, C. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. C.A.R.Hoare Series in Computer Science. Prentice Hall International, 1993.

[143] B. Kernighan and D. Ritchie. *C Programming Language*. Prentice Hall, 2nd edition, 1989.

[144] D. Knuth. *The Art of Computer Programming, Vol.1: Fundamental Algorithms*. Addison-Wesley, Reading, Mass., USA, 1968.

[145] D. Knuth. *The Art of Computer Programming, Vol.2.: Seminumerical Algorithms*. Addison-Wesley, Reading, Mass., USA, 1969.

[146] D. Knuth. *The Art of Computer Programming, Vol.3: Searching & Sorting*. Addison-Wesley, Reading, Mass., USA, 1973.

[147] I. Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery (Eds.: J. Worrall and E. G. Zahar)*. Cambridge University Press, The Edinburgh Building, Shaftesbury Road, Cambridge CB2 2RU, England, 2 September 1976. ISBN: 0521290384. Published in 1963-64 in four parts in the British Journal for Philosophy of Science. (Originally Lakatos' name was Imre Lipschitz.).

[148] L. Lamport. *Specifying Systems*. Addison–Wesley, Boston, Mass., USA, 2002.

[149] A. Lamsweerde. *Requirements Engineering: from system goals to UML models to software specifications*. Wiley, 2009.

[150] P. Landin. Histories of discoveries of continuations: Belles-lettres with equivocal tenses, 1997. In O. Danvy, editor, ACM SIGPLAN Workshop on Continuations, Number NS-96-13 in BRICS Notes Series, 1997.

[151] P. J. Landin. A Correspondence Between ALGOL 60 and Church's Lambda-Notation (in 2 parts). *Communications of the ACM*, 8(2-3):89–101 and 158–165, Feb.-March 1965.

[152] P. J. Landin. A Generalization of Jumps and Labels. Technical report, Univac Sys. Prgr. Res. Grp., N.Y., 1965.

[153] P. J. Landin. Getting Rid of Labels. Technical report, Univac Sys. Prgr. Res. Grp., N.Y., 1965.

[154] J. Lee. *Computer Semantics*. Van Nostrand Reinhold Co., 1972.

[155] J. Lee and W. Delmore. The Vienna Definition Language, a generalization of instruction definitions. In *SIGPLAN Symp. on Programming Language Definitions, San Francisco*, Aug. 1969.

[156] H. S. Leonard and N. Goodman. The Calculus of Individuals and its Uses. *Journal of Symbolic Logic*, 5:45–44, 1940.

[157] X. Leroy and P. Weis. *Manuel de Référence du langage Caml*. InterEditions, Paris, France, 1993. ISBN 2-7296-0492-8.

[158] T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. Addison-Wesley & Sun Microsystems. ACM Press Books, 1996. 496 pp, ISBN 0-10-63452-X.

[159] J. Lipson. *Elements of Algebra and Algebraic Computing*. Addison-Wesley, Reading, Mass., 1981.

[160] W. Little, H. Fowler, J. Coulson, and C. Onions. *The Shorter Oxford English Dictionary on Historical Principles*. Clarendon Press, Oxford, England, 1987.

[161] J. Lloyd. *Foundation of Logic Programming*. Springer-Verlag, 1984.

[162] P. Lucas. Formal Semantics of Programming Languages: VDL. *IBM Journal of Devt. and Res.*, 25(5):549–561, 1981.

[163] P. Lucas and K. Walk. On the Formal Description of PL/I. *Annual Review Automatic Programming Part 3*, 6(3), 1969.

[164] E. Luschei. *The Logical Systems of Leśniewksi*. North Holland, Amsterdam, The Netherlands, 1962.

[165] ANSI X3.23-1974. The Cobol programming language. Technical report, American National Standards Institute, Standards on Computers and Information Processing, 1974.

[166] ANSI X3.53-1976. The PL/I programming language. Technical report, American National Standards Institute, Standards on Computers and Information Processing, 1976.

[167] ANSI X3.9-1966. The Fortran programming language. Technical report, American National Standards Institute, Standards on Computers and Information Processing, 1966.

[168] J. McCarthy. A Basis for a Mathematical Theory of Computation. In *Computer Programming and Formal Systems*. North-Holland Publ.Co., Amsterdam, 1963.

[169] J. McCarthy. Artificial Intellingence. Electronically, on the Web: `http://www-formal.-stanford.edu/jmc/`, 2004.

[170] J. McCarthy and et al. *LISP 1.5, Programmer's Manual*. The MIT Press, Cambridge, Mass., USA, 1962.

[171] S. Merz. On the Logic of TLA+. *Computing and Informatics*, 22(1–2), 2003.

[172] J. Meseguer. Software Specification and Verification in Rewriting Logic. NATO Advanced Study Institute, 2003.

[173] Microsoft Corporation. *MCAD/MCSD Self-Paced Training Kit: Developing Web Applications with Microsoft Visual Basic .NET and Microsoft Visual C# .NET*. Microsoft Corporation, Redmond, WA, USA, 2002. 800 pages.

[174] Microsoft Corporation. *MCAD/MCSD Self-Paced Training Kit: Developing Windows-Based Applications with Microsoft Visual Basic .NET and Microsoft Visual C# .NET*. Microsoft Corporation, Redmond, WA, USA, 2002.

[175] D. Miéville and D. Vernant. *Stanisław Leśniewksi aujourd'hui*. Grenoble, October 8-10, 1992.

[176] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML*. The MIT Press, Cambridge, Mass., USA and London, England, 1990.

[177] C. C. Morgan. *Programming from Specifications*. International Series in Computer Science. Prentice Hall, Hemel Hempstead, Hertfordshire HP2 4RG, UK, 1990.

[178] T. Mossakowski, A. E. Haxthausen, D. Sanella, and A. Tarlecki. CASL — The Common Algebraic Specification Language: Semantics and Proof Theory. *Computing and Informatics*, 22(1–2), 2003.

[179] J. F. Nilsson. Some Foundational Issues in Ontological Engineering, October 30 – Novewmber 1 2002. Lecture slides for a PhD Course in *Representation Formalisms for Ontologies*, Copenhagen, Denmark.

[180] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL, A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[181] Object Management Group. *OMG Unified Modelling Language Specification*. OMG/UML, http://www.omg.org/uml/, version 1.5 edition, March 2003. www.omg.org/cgi-bin/doc?formal/03-03-01.

[182] E.-R. Olderog and H. Dierks. *Real-Time Systems: Formal Specification and Automatic Verification*. Cambridge University Press, UK, 2008.

[183] O. Ore. *Graphs and their Uses* . The Mathematical Association of America, 1963.

[184] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS Language Reference*. Computer Science Laboratory, SRI International, Menlo Park, CA, Sept. 1999.

[185] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS System Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, Sept. 1999.

[186] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, Dec. 1972.

[187] D. L. Parnas. A technique for software module specification with examples. *Communications of the ACM*, 14(5), May 1972.

[188] D. L. Parnas. *Software Fundamentals: Collected Papers, Eds.: David M. Weiss and Daniel M. Hoffmann*. Addison–Wesley Publ. Co., April 9 2001. 688 pages. ISBN 0201703696. Amazon price (August 2001) US˙$ 49.95.

[189] D. L. Parnas and P. C. Clements. A rational design process: How and why to fake it. *IEEE Trans. Software Engineering*, 12(2):251–257, February 1986.

[190] D. L. Parnas, P. C. Clements, and D. M. Weiss. Enhancing reusability with information hiding. In *Tutorial: Software Reusability (Ed.: Peter Freeman)*, pages 83–90. IEEE Press, 1986.

[191] L. Paulson. Isabelle: The next 700 theorem provers. In P. Oddifreddi, editor, *Logic in Computer Science*, pages 361–386. Academic Press, 1990.

[192] C. Petzold. *Programming Windows with C# (Core Reference)* . Microsoft Corporation, Redmond, WA, USA, 2001. 1200 pages.

[193] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical report, Comp. Sci. Dept., Aarhus Univ., Denmark; DAIMI-FN-19, 1981. Definitive version of this seminal report is (to be) published in a special issue of the *Journal of Logic and Algebraic Programming* (eds. Jan Bergstra and John Tucker) devoted to a workshop on SOS: Structural Operational Semantics, a Satellite Event of CONCUR 2004, August 30, 2004, London, United Kingdom. Look also for Gordon Plotkin's introductory paper for that issue: The Origins of Structural Operational Semantics.

[194] G. D. Plotkin. Structural operational semantics. Lecture notes, Aarhus University, DAIMI FN-19. Reprinted 1991, 1981. See [196, 195].

[195] G. D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:3–15, July-December 2004. See [194, 196].

[196] G. D. Plotkin. A structural approach operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:17–139, July-December 2004. Widely disseminated since 1981 as [194]. See also [195].

[197] B. Randell. On Failures and Faults. In *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 18–39. Formal Methods Europe, Springer–Verlag, 2003. Invite Paper.

[198] W. Reisig. On Gurevich's Theorem for Sequential Algorithms. *Acta Informatica*, 2003.

[199] W. Reisig. The Expressive Power of Abstract State Machines. *Computing and Informatics*, 22(1–2), 2003. This paper is one of a series: [60, 81, 178, 111, 171, 126] appearing in a double issue of the same journal: *Logics of Specification Languages* — edited by Dines Bjørner.

[200] W. Reisig. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Leitfäden der Informatik. Vieweg+Teubner, 1st edition, 15 June 2010. 248 pages; ISBN 978-3-8348-1290-2.

[201] J. C. Reynolds. *The Craft of Programming*. Prentice-Hall, 1981.

[202] J. C. Reynolds. *The Semantics of Programming Languages*. Cambridge University Press, 1999.

[203] A. W. Roscoe. *Theory and Practice of Concurrency*. C.A.R. Hoare Series in Computer Science. Prentice-Hall, 1997. Now available on the net: http://www.comlab.ox.ac.uk/-people/bill.roscoe/publications/68b.pdf.

[204] B. Russell. The Philosophy of Logical Atomism. *The Monist: An International Quarterly Journal of General Philosophical Inquiry,*, xxxviii–xxix:495–527, 32–63, 190–222, 345–380, 1918–1919.

[205] B. Russell. *Introduction to Mathematical Philosophy*. George Allen and Unwin, London, 1919.

[206] D. A. Schmidt. *Denotational Semantics: a Methodology for Language Development*. Allyn & Bacon, 1986.

[207] S. Schneider. *Concurrent and Real-time Systems — The CSP Approach*. Worldwide Series in Computer Science. John Wiley & Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UD, England, January 2000.

[208] P. Sestoft. *Java Precisely*. The MIT Press, 25 July 2002. 100 pages (sic !), ISBN 0262692767.

[209] P. M. Simons. *Foundations of Logic and Linguistics: Problems and their Solutions*, chapter Leśniewski's Logic and its Relation to Classical and Free Logics. Plenum Press, New York, 1985. Georg Dorn and P. Weingartner (Eds.).

[210] J. M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*, volume 3 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Jan. 1988.

[211] J. M. Spivey. *The Z Notation: A Reference Manual*. International Series in Computer Science. Prentice Hall, Hemel Hempstead, Hertfordshire HP2 4RG, UK, 1989.

[212] J. Srzednicki and Z. Stachniak, editors. *Leśniewksi's Lecture Notes in Logic*. Dordrecht, 1988.

[213] J. Srzednicki and Z. Stachniak. *Leśniewksi's Systems Protothetic*. . Dordrecht, 1998.

[214] Staff of Merriam Webster. Online Dictionary: `http://www.m-w.com/home.htm`, 2004. Merriam–Webster, Inc., 47 Federal Street, P.O. Box 281, Springfield, MA 01102, USA.

[215] J. Stein, editor. *The Random House American Everyday Disctionary*. Random House, New York, N.Y., USA, 1949, 1961.

[216] B. Stroustrup. *C++ Programming Language*. Addison-Wesley Publishing Company, 1986.

[217] S. J. Surma, J. T. Srzednicki, D. I. Barnett, and V. F. Rickey, editors. *Stanisław Leśniewksi: Collected works (2 Vols.)*. Dordrecht, Boston – New York, 1988.

[218] T. Tamai. Social Impact of Information System Failures. *Computer, IEEE Computer Society Journal*, 42(6):58–65, June 2009.

[219] R. Tennent. *The Semantics of Programming Languages*. Prentice–Hall Intl., 1997.

[220] S. Thompson. *Haskell: The Craft of Functional Programming*. Addison Wesley, 2nd edition, 29 March 1999. 512 pages, ISBN 0201342758.

[221] D. Turner. Miranda: A Non-strict Functional Language with Polymorphic Types. In J. Jouannaud, editor, *Functional Programming Languages and Computer Architectures*, number 201 in Lecture Notes in Computer Science. Springer-Verlag, 1985.

[222] J. van Benthem. *The Logic of Time*, volume 156 of *Synthese Library: Studies in Epistemology, Logic, Methhodology, and Philosophy of Science (Editor: Jaakko Hintika)*. Kluwer Academic Publishers, P.O.Box 17, NL 3300 AA Dordrecht, The Netherlands, second edition, 1983, 1991.

[223] R. van Glabbeek and P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. Electronically, on the Web: `http://theory.stanford.edu/~rvg/abstraction/abstraction.html`, Centrum voor Wiskunde en Informatica, Postbus 94079, 1090 GB Amsterdam, The Netherlands, January 1996.

[224] A. van Wijngaarden. Report on the algorithmic language Algol 68. *Acta Informatica*, 5:1–236, 1975.

[225] B. Venners. *Inside the Java 2.0 Virtual Machine (Enterprise Computing)*. McGraw-Hill; ISBN: 0071350934, October 1999.

[226] D. Watt, B. Wichmann, and W. Findlay. *Ada: Language and Methodology*. Intl. Ser. in Comp. Sc. Prentice-Hall International, 1986.

[227] P. Weis and X. Leroy. *Le langage Caml*. Dunod, Paris, France, 1999. ISBN 2-10-004383-8, Second edition.

[228] Wikipedia. Polymorphism. In *Internet*. Published: http://en.wikipedia.org/wiki/Polymorphism_(computer_science), 2005.

[229] G. Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, Cambridge, Mass., USA, 1993.

[230] N. Wirth. The Programming Language PASCAL. *Acta Informatica*, 1(1):35–63, 1971.

[231] N. Wirth. *Systematic Programming*. Prentice-Hall, 1973.

[232] N. Wirth. *Algorithms + Data Structures = Programs*. Prentice-Hall, 1976.

[233] N. Wirth and C. Hoare. A Contribution to the Development of ALGOL. *Communications of the ACM*, 9(6):413–432, 1966.

[234] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Proof and Refinement*. Prentice Hall International Series in Computer Science, 1996.

[235] E. N. Zalta. Logic. In *The Stanford Encyclopedia of Philosophy*. Published: http://plato.stanford.edu/, Winter 2003.

[236] C. C. Zhou and M. R. Hansen. *Duration Calculus: A Formal Approach to Real–time Systems*. Monographs in Theoretical Computer Science. An EATCS Series. Springer–Verlag, 2004.