

# Modeling with Abstract State Machines: A support for accurate system design and analysis

Egon Börger  
Computer Science Department,  
University of Pisa,  
I-56125 Pisa,  
boerger@di.unipi.it

**Abstract:** We survey applications of the Abstract State Machines (ASM) method for high-level system modeling and for well-documented refinements of abstract models to code.

The outstanding feature of the ASM method is that within a single, precise yet simple, conceptual framework it naturally supports and uniformly links the major activities which occur during the typical software life cycle, namely:

- requirements capture by constructing satisfactory ground models, i.e. accurate high-level system blueprints, serving as precise contract and formulated in a language which is understood by all stakeholders (see [Bö03a]),
- detailed design by stepwise refinement, bridging the gap between specification and code design by piecemeal, systematically documented detailing of abstract models down to executable code (see [Bö03b]),
- validation of models by their simulation, based upon the notion of ASM run and supported by numerous tools to execute ASMs (ASM Workbench [De01], Asm-Gofer [Sc], C-based XASM [AK01], .NET-executable AsmL engine [Fo01]),
- verification of model properties by proof techniques, also tool supported, e.g. by KIV [SA97] or PVS [Do98, GR00] or model checkers [Wi97, DW00, GTW03],
- documentation for inspection, reuse and maintenance by providing, through the intermediate models and their analysis, explicit descriptions of the software structure and of the major design decisions.

The key to the practicability of ASMs also under industrial constraints is to be found in the simple and intuitive way in which ASMs support defining most general Virtual Machines (VMs) and their refinements to lower levels of abstraction. ASMs naturally extend Finite State Machines by allowing a) states with arbitrarily complex or abstract data structures and b) runs with transitions where multiple components act either simultaneously (synchronous parallelism) or asynchronously (like globally asynchronous, locally

synchronous Codesign-FSMs). The flexible ASM refinement notion provides a uniform conceptual framework for effectively relating different system views and aspects, in both design and analysis, filling a gap in the UML framework.

The general yet frugal character of the ASM language, namely for a rigorous form of pseudo-code over abstract data, is the source for the flexibility in modeling and analysis which has been experienced in such different areas as:

- industrial standardization projects: models for the ISO-Prolog standard [BD90], the ECMA standard for C# [BFGS04], the IEEE-VHDL93 standard [BGM94], the ITU-T standard for SDL-2000 [GGP03],
- programming languages: definition and analysis of the semantics and the implementation for the major real-life programming languages, e.g. SystemC [MRR03], Java and its implementation on the Java Virtual Machine [SSB01], domain-specific languages used at the Union Bank of Switzerland [KST98], etc.
- architectural design: verification (e.g. of pipelining schemes [BM97] or of VHDL-based hardware design at Siemens [Sc02, Ch.2]), architecture and compiler co-exploration [Te01, TWFT00],
- reengineering and design of industrial control systems: software projects at Siemens related to railway [BBC<sup>+</sup>96, BPS00] and mobile telephony network components [CP02], debugger specification at Microsoft [BBG<sup>+</sup>00],
- protocols: for authentication, cryptography, cache-coherence, routing-layers for distributed mobile ad hoc networks, group-membership etc., focussed on verification,
- verification of compilation schemes and compiler back-ends [BR95, BD96, DGVZ98, SSB01],
- modeling e-commerce [AVFY98] and web services [FGV04],
- simulation and testing: fire detection system in coal mines [BCF<sup>+</sup>96], simulation of railway scenarios at Siemens [BPS00], implementation of behavioral interface specifications on the .NET platform and conformance test of COM components at Microsoft [BS02], compiler testing [KKP<sup>+</sup>03], test case generation [GR03].

The AsmBook [BS03] introduces into the ASM method and illustrates it by textbook examples, which are extracted from real-life case studies and industrial applications. Additional material, including slide decks for lecturers, can be downloaded from the AsmBook website [www.di.unipi.it/AsmBook/](http://www.di.unipi.it/AsmBook/).

## References

- [AK01] Anlauff, M. und Kutter, P. Xasm Open Source. Web pages at <http://www.xasm.org/>. 2001.

- [AVFY98] Abiteboul, S., Vianu, V., Fordham, B., und Yesha, Y.: Relational transducers for electronic commerce. In: *Proc. 17th ACM Sympos. Principles of Database Systems (PODS 1998)*. S. 179–187. ACM Press. 1998.
- [BBC<sup>+</sup>96] Börger, E., Busch, H., Cuellar, J., Pöppinghaus, P., Tiden, E., und Wildgruber, I. Konzept einer hierarchischen Erweiterung von EURIS. Siemens ZFE T SE 1 Internal Report BBCPTW91-1 (pp. 1–43). Summer 1996.
- [BBG<sup>+</sup>00] Barnett, M., Börger, E., Gurevich, Y., Schulte, W., und Veanes, M.: Using Abstract State Machines at Microsoft: A case study. In: Gurevich, Y., Kutter, P., Odersky, M., und Thiele, L. (Hrsg.), *Abstract State Machines: Theory and Applications*. volume 1912 of *Lecture Notes in Computer Science*. S. 367–380. Springer-Verlag. 2000.
- [BCF<sup>+</sup>96] Burgard, W., Cremers, A. B., Fox, D., Heidelbach, M., Kappel, A. M., und Lüttringhaus-Kappel, S.: Knowledge-enhanced CO-monitoring in coal mines. In: *Proc. Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE)*. S. 511–521. Fukuoka, Japan. 4–7 June 1996. .
- [BD90] Börger, E. und Dässler, K.: Prolog: DIN papers for discussion. ISO/IEC JTC1 SC22 WG17 Prolog Standardization Document 58. National Physical Laboratory. Middlesex, England. 1990.
- [BD96] Börger, E. und Durdanović, I.: Correctness of compiling Occam to Transputer code. *Computer Journal*. 39(1):52–92. 1996.
- [BFGS04] Börger, E., Fruja, G., Gervasi, V., und St'ark, R.: A complete formal definition of the semantics of C#. *Theoretical Computer Science*. to appear. 2004.
- [BGM94] Börger, E., Glässer, U., und Müller, W.: The semantics of behavioral VHDL'93 descriptions. In: *EURO-DAC'94. European Design Automation Conference with EURO-VHDL'94*. S. 500–505. Los Alamitos, California. 1994. IEEE Computer Society Press.
- [BM97] Börger, E. und Mazzanti, S.: A practical method for rigorously controllable hardware design. In: Bowen, J. P., Hinchey, M. B., und Till, D. (Hrsg.), *ZUM'97: The Z Formal Specification Notation*. volume 1212 of *LNCS*. S. 151–187. Springer-Verlag. 1997.
- [B'03a] Börger, E.: The ASM ground model method as a foundation of requirements engineering. In: N.Dershowitz (Hrsg.), *Manna-Symposium*. volume 2772 of *LNCS*. Springer-Verlag. 2003.
- [B'03b] Börger, E.: The ASM refinement method. *Formal Aspects of Computing*. 15:237–257. 2003.
- [BPS00] Börger, E., Pöppinghaus, P., und Schmid, J.: Report on a practical application of ASMs in software design. In: Gurevich, Y., Kutter, P., Odersky, M., und Thiele, L. (Hrsg.), *Abstract State Machines: Theory and Applications*. volume 1912 of *LNCS*. S. 361–366. Springer-Verlag. 2000.
- [BR95] Börger, E. und Rosenzweig, D.: The WAM – definition and compiler correctness. In: Beierle, C. und Plümer, L. (Hrsg.), *Logic Programming: Formal Methods and Practical Applications*. volume 11 of *Studies in Computer Science and Artificial Intelligence*. chapter 2, S. 20–90. North-Holland. 1995.
- [BS02] Barnett, M. und Schulte, W.: Contracts, components and their runtime verification on the .NET platform. *J. Systems and Software*. Special Issue on Component-Based Software Engineering. 2002.

- [BS03] Börgler, E. und Stärk, R. F.: *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer. 2003.
- [CP02] Castillo, G. D. und Poppinghaus, P.: Designing software for internet telephony: experiences in an industrial development process. In: Blass, A., Börgler, E., und Gurevich, Y. (Hrsg.), *Theory and Applications of Abstract State Machines*. Schloss Dagstuhl, Int. Conf. and Research Center for Computer Science. 2002.
- [De01] Del Castillo, G.: *The ASM Workbench. A Tool Environment for Computer-Aided Analysis and Validation of Abstract State Machine Models*. PhD thesis. Universität Paderborn. Germany. 2001.
- [DGVZ98] Dold, A., Gaul, T., Vialard, V., und Zimmermann, W.: ASM-based mechanized verification of compiler back-ends. In: Glässer, U. und Schmitt, P. (Hrsg.), *Proc. 5th Int. Workshop on Abstract State Machines*. S. 50–67. Magdeburg University. 1998.
- [Do98] Dold, A.: A formal representation of Abstract State Machines using PVS. Verifix Technical Report Ulm/6.2. Universität Ulm. Germany. July 1998.
- [DW00] Del Castillo, G. und Winter, K.: Model checking support for the ASM high-level language. In: Graf, S. und Schwartzbach, M. (Hrsg.), *Proc. 6th Int. Conf. TACAS 2000*. volume 1785 of *Lecture Notes in Computer Science*. S. 331–346. Springer-Verlag. 2000.
- [FGV04] Farahbod, R., Glässer, U., und Vajihollahi, M.: Specification and validation of the business process execution language for web services. In: Thalheim, B. und Zimmermann, W. (Hrsg.), *Abstract State Machines 2004*. Lecture Notes in Computer Science. Springer-Verlag. 2004.
- [Fo01] Foundations of Software Engineering Group, Microsoft Research. AsmL. Web pages at <http://research.microsoft.com/foundations/AsmL/>. 2001.
- [GGP03] Glässer, U., Gotzhein, R., und Prinz, A.: Formal semantics of sdl-2000: Status and perspectives. *Computer Networks*. 42(3):343–358. June 2003.
- [GR00] Gargantini, A. und Riccobene, E.: Encoding Abstract State Machines in PVS. In: Gurevich, Y., Kutter, P., Odersky, M., und Thiele, L. (Hrsg.), *Abstract State Machines: Theory and Applications*. volume 1912 of *LNCS*. S. 303–322. Springer-Verlag. 2000.
- [GR03] Gargantini, A. und Riccobene, E.: Using Spin to generate tests from ASM specifications. In: Börgler, E., Gargantini, A., und Riccobene, E. (Hrsg.), *Abstract State Machines 2003–Advances in Theory and Applications*. volume 2589 of *Lecture Notes in Computer Science*. S. 263–277. Springer-Verlag. 2003.
- [GTW03] Gawanmeh, A., Tahar, S., und Winter, K.: Interfacing ASMs with the MDG tool. In: Börgler, E., Gargantini, A., und Riccobene, E. (Hrsg.), *Abstract State Machines 2003–Advances in Theory and Applications*. volume 2589 of *Lecture Notes in Computer Science*. S. 278–292. Springer-Verlag. 2003.
- [KKP<sup>+</sup>03] Kalinov, A., Kossatchev, A., Petrenko, A., Posypkin, M., und Shishkov, V.: Using ASM specifications for compiler testing. In: Börgler, E., Gargantini, A., und Riccobene, E. (Hrsg.), *Abstract State Machines 2003–Advances in Theory and Applications*. volume 2589 of *Lecture Notes in Computer Science*. S. 415. Springer-Verlag. 2003.
- [KST98] Kutter, P., Schweizer, D., und Thiele, L.: Integrating domain specific language design in the software life cycle. In: *Proc. Int. Workshop on Current Trends in Applied Formal Methods*. volume 1641 of *Lecture Notes in Computer Science*. S. 196–212. Springer-Verlag. 1998.

- [MRR03] Mueller, W., Ruf, J., und Rosenstiel, W.: An ASM-based semantics of systemC simulation. In: Mueller, W., Ruf, J., und Rosenstiel, W. (Hrsg.), *SystemC - Methodologies and Applications*. S. 97–126. Kluwer Academic Publishers. 2003.
- [SA97] Schellhorn, G. und Ahrendt, W.: Reasoning about Abstract State Machines: The WAM case study. *J. Universal Computer Science*. 3(4):377–413. 1997.
- [Sc] Schmid, J. Executing ASM specifications with AsmGofer. Web pages at <http://www.tydo.de/AsmGofer>.
- [Sc02] Schmid, J.: *Refinement and Implementation Techniques for Abstract State Machines*. PhD thesis. University of Ulm. Germany. 2002.
- [SSB01] Stärk, R. F., Schmid, J., und Börgler, E.: *Java and the Java Virtual Machine: Definition, Verification, Validation*. Springer-Verlag. 2001. .
- [Te01] Teich, J. Project Buildabong at University of Paderborn. <http://www-date.upb.de/RESEARCH/BUILDABONG/buildabong.html>. 2001.
- [TWFT00] Teich, J., Weper, R., Fischer, D., und Trinkert, S.: A joint architecture/compiler design environment for ASIPs. In: *Proc. Int. Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES2000)*. S. 26–33. San Jose, CA, USA. November 2000. ACM Press.
- [Wi97] Winter, K.: Model checking for Abstract State Machines. *J. Universal Computer Science*. 3(5):689–701. 1997.