Journal of Universal Computer Science, vol. 14, no. 12 (2008), 1921-1928
submitted: 16/6/08, accepted: 23/6/08, appeared: 28/6/08

Quo Vadis Abstract State Machines?

J.UCS Special Issue

Egon Börger
(Università di Pisa, Italy
boerger@di.unipi.it)

Andreas Prinz
(University of Grimstadt, Norway)
andreas.prinz@uia.no)

Abstract: In introducing this special ASM issue of J.UCS we point out the particular role this Journal played in the short history of the ASM method and add some reflections on its current status.

1 J.UCS and the ASM Method

The *Journal of Universal Computer Science* has marked some important steps of the development of the Abstract State Machines (ASM) method for a rigorous design and the mathematical analysis of complex computer-based systems. It is the first journal which devoted a special issue to the ASM method and its applications, and within the last decade this is the fourth special ASM issue hosted by J.UCS. In addition, some other influential papers on the ASM method have been published as single papers in J.UCS.

The first ASM issue in J.UCS (3.4 in 1997) dealt with ASM theory: foundational questions, questions from complexity theory and logic, investigation of the central notion of refinement and of machine support for reasoning about ASMs. The second ASM issue in J.UCS (3.5 in 1997) was devoted to applications of ASMs to classical problems of programming and software engineering: semantics of programming languages, compiler correctness (verifiable design and implementation of real-life programming languages), integrating ASMs into the software development life cycle. The third ASM issue in J.UCS (7.11 in 2001) was the first journal issue to publish revised and rereviewed versions of selected best papers from an *International ASM Workshop*, to be specific the 8th one, held in 2001 in Las Palmas (Gran Canaria). The themes of J.UCS 7.11 in 2001 covered the full range from theoretical foundations to industrial applications, as is typical for ASM workshops since their establishment. Also the individual papers that appeared in this journal span from theory to applications: mathematical investigation of the Kerberos protocol [BR97], a formal definition of the International Telecommunication Union standard for SDL [GK97], an ASM-based software development method leading from rigorously modeling informal requirements to compilable code with verifiable properties, illustrated for the light control case study [BRS00], an account of the development of the ASM method [Bör02].

After the ASM issue in J.UCS in 2001, two more editions of the *International ASM Workshop* series saw the publication of revised selected best papers in special journal issues. The 10th edition, which took place in 2003 in Taormina (Sicily), aimed at an integration of ASM-based modeling, validation and verification techniques into neighboring system engineering methods, and was documented in [BGR03] and the special ASM issue in the *Theoretical Computer Science* journal (336 (2-3), 2005). The 12th edition, which took place in 2005 in Paris (France), was dedicated to mathematical techniques and their implementation for ASM-based system design and analysis and was documented in the special ASM issue in the *Fundamenta Informaticae* journal [BS07].

The J.UCS issue presented here contains selected best papers from the 14th International ASM Workshop, which took place in Grimstadt, Norway, in June 2007 and was characterized by contributions ranging from theory to industrial applications. After the workshop we invited the speakers to submit during the Summer of 2007 full papers to this special issue. As a result of a double reviewing procedure[1] the reader finds seven papers in this issue. One paper (see [AFL08]) surveys recent industrial applications of the ASM method in the area of web services (their mediation, discovery and composition). One paper (see [BB08]) illustrates how to integrate the ASM method into a feature-based software engineering discipline to modularize proofs of properties for software product lines. Three papers ([GRS08, OL08, SV08]) contribute to the tool environment for executing ASMs from certain classes, notably ASMs including the consideration of resources and of real-time properties. One paper (see [Bel08]) surveys security issues and was invited to trigger applications of the ASM method in the field. One paper (see [Sch08]) is on recent progress for the ASM refinement notion that resulted from its recent application to the verification of the well-known Mondex protocol.

2 Present Status and Some Challenges

2.1 Present Status

The ASM method by now is an established mathematics-based and industrially successful method for an accurate design of complex computer-based systems

[1] A first round of reviews lasted from September until the end of 2007 and resulted in the selection of the papers present in this volume, which were accepted for a revision reflecting the criticism of the reviews and a resubmission by April 30 (2008). The second review round ended at the end of May 2008.

that is linkable to a precise analysis. The analysis covers both experimental validation (by simulation) and mathematical verification (by manual or machine assisted or automated proofs or by model checking). The validation of ASMs exploits the fact that they are not logical formulae but machines and as such come with a notion of run (execution). The verification of ASMs exploits the fact that they come with a (simple) mathematical definition of their semantics and of a practial refinement notion, whereby they can be made subject to far reaching system decompositions with precise formulations of the interfaces and composable proofs of the intended properties. The design capabilities of the ASM method also cover capturing requirements by ASM models in a reliable way. Reliability means that the appropriateness of the models can be checked by the application domain experts, the persons who are responsible for the requirements, and can be used by the system developers for a stepwise detailing (by provably controllable ASM refinement steps) to executable code. This exploits the abstraction potential of ASMs, which is unconstrained by any formal straitjacket. The reader who is interested in detailed information and references on the achievements of the ASM method may read the historical account in this Journal [Bör02] (complete as of 2002). Not surprisingly the ASM method is best qualified (see [Bör08b, Bör07]) for being used in the challenging Verified Software Initiative, which is kicked off during these days by the *Second IFIP Working Conference on Verified Software: Theories, Tools, and Experiments* (VSTTE 2008), "a fifteen-year, cooperative, international project directed at the scientific challenges of large-scale software verification" (quoted from http://qpq.csl.sri.com/vsr/vstte-08, see also [MW08]).

An interesting phenomenon, which confirms the maturity and practicality of the ASM method, is that over the last five years numerous publications with ASM-based research results appeared in a variety of journals and conferences spread over the field of computer science and its applications and are not any more presented at the (still regularly held) ASM workshops. So the question is whether there is still something to be done for the ASM approach to system development. We believe yes. Since this is not the place for an extensive proposal of future research themes, we restrict ourselves here to list some issues we consider as worth to be investigated within the ASM method.

2.2 Some Challenges

Concerning computational concepts: asynchronous (also called distributed) ASMs badly need further development. For example, we need practically useful patterns for communication and synchronization of multi-agent ASMs, in particular supporting omnipresent calling structures (like RPC, RMI and related middleware constructs) and web service interaction patterns.[2] Furthermore the concept of time, which for synchronous (also called sequential) ASMs coincides with the order of natural numbers, has not yet received a practically viable foundation for multi-agent ASMs. For the real-time case some steps are made in the two papers [OL08, SV08] in this issue.

Concerning modeling aspects: The ASM method badly needs to be integrated into current modeling and software engineering practice, both conceptually and concerning tool support (see below). By conceptual integration we mean to provide ASM libraries where rigorous ASM-based definitions are offered for basic method patterns as they are used in current practical frameworks. An example are behavioral definitions of architectural (de)composition techniques, in particular for dynamic Web application architectures. Another example are rigorous semantical definitions for various successful notations as used in current system design and programming practice, including diagram-based graphical notations. This is feasible, as one can see from the accurate semantical models that have been developed in terms of ASMs for numerous representative programming or modeling languages, including pictorial notations like the ones in UML [Cav00, CRS03, Obe03], SDL [EGG$^+$01], BPEL [FGV04, RFV06] and BPMN [BT08]. Such ASM libraries should support porting application programs in a coherent way between different platforms or languages, due to the "codeless" form of programming represented by building ASM models. This challenge has also a verification aspect, namely to combine feature-based modular *design* and *proof* techniques, with the goal to scale verification to software product lines and to integrate it into current software development practice, as is advocated in the paper [BB08] in this issue. The combination of modeling and verification is a theme that relates the ASM method to Abrial's B-method [Abr96], which stresses the point of (and provides strong support for) computer-assisted verification.

A less technical but nevertheless very important desideratum, to support the modeling activities with ASMs, is to develop precise pragmatical guidelines explaining how to apply the concepts and techniques offered by the ASM method when building rigorous models leading from informal requirements to compilable code. One subproblem probably worth some investigation concerns methods supporting the extraction of ground model elements from natural language descriptions of requirements. Inspiration should be taken from established process models, e.g. the Rational Unified Process [Kru03].

Concerning tool support: a great variety of experiments has been made with different tools, in particular for executing classes of ASMs. Each such simulator

[2] Theoretical ASM-based interaction schemes, so-called interactive small-step algorithms made up for proofs of various parallel ASM theses, have been analyzed in [BG07, BGRR06]. The set of practical interaction patterns proposed in [AMH05] has been modeled by ASMs in [BB05].

had its raison d'etre, created with a specific goal in mind, and has served its immediate purpose it had been built for, see [BS03, Ch.9.4.3] for a survey of the numerous tools built until 2003; for recent tools see [F^+, FGG06] and the three papers [GRS08, OL08, SV08] in this issue. For verifying properties of ASMs leading existing theorem provers and model-checkers have been linked to ASMs. What we need now is an *integrated environment* of cooperating (not isolated) tools for design, validation and verification, and its deployment for industrial applications. This environment needs to support the different activities of defining, transforming (by refinements, including code generation) and analysing ASM models: by testing, via simulation that is supported by good visualization and debugging mechanisms, and by verification. The tool environment has to enable us to capture the design knowledge in a rigorous, electronically available and reusable way, and to achieve this goal it must be integrated into established design flows and their standard tool environments.

Progress in this direction can be expected from a cooperation with neighbouring modeling and verification approaches, in particular the B-method [Abr96]. A good start in this direction has been made by organizing joint meetings. The first one was the 2006 Dagstuhl Seminar on *Rigorous Methods for Software Construction and Analysis*[3]. It is followed this Fall by the ABZ 2008 Conference[4] in London, where two tutorials on ASM and B tools are given. Since Event-B systems can be viewed in a natural way as particular classes of ASMs (see [Bör08a, Sect.6.1]), it should be possible to strongly relate the two methods and link their tools, to the advantage of both.

Acknowledgements

We thank the seven members of the program committee of the International ASM Workshop ASM'2007 for assisting us in reviewing the submitted papers for presentation at the workshop, which constituted a first selection step: Uwe Glässer (Simon Fraser University, Canada), Yuri Gurevich (Microsoft Research, USA), Elvinia Riccobene (University of Milan, Italy), Bernhard Thalheim (Christian Albrechts University, Germany), Margus Veanes (Microsoft Research, USA), Charles Wallace (Michigan Technological University, USA), Wolf Zimmermann (Martin Luther University, Germany).

We thank the 31 colleagues who acted as additional reviewers in the second and third phase of the procedure for carefully reviewing the submissions and resubmissions of the full papers for this J.UCS issue: David Aspinall (U. Edinburgh), Richard Banach (U. Manchester), Andreas Blass (U. Michigan at Ann Arbor), Antonio Brogi (U. Pisa), Pietro Cenciarelli (U. Roma), Massimo

[3] http://www.dagstuhl.de/06191/ and [AG08]
[4] http://www.abz2008.org/ and [BBBB08]

Coppola (U. Pisa), Roozbeh Farahbod (Simon Fraser U. Vancouver), Joachim Fischer (Humboldt U. Berlin), Leo Freitas (U. York), Angelo Gargantini (U. Bergamo), Andreas Glausch (Humboldt U. Berlin), Mats Heimdahl (U. Minnesota), Jim Huggins (Kettering U.), Kai Koskimies (Tampere U. of Technology), Igor Kotenko (St. Petersburg Institute for Informatics and Automation), Birger Moeller-Pedersen (U. Oslo), Wolfgang Müller (U. Paderborn), Zsolt Nemeth (U. Budapest), Rolf Nossum (U. Agder), Ileana Ober (U. Toulouse), Vladimir Oleshchuk (U. Agder), Wolfgang Reisig (Humboldt U. Berlin), Peter Schmitt (U. Karlsruhe), Jaroslav Sevcik (U. Edinburgh), Anatol Slissenko (U. Paris 7), Sofiene Tahar (U. Montreal), Mark Utting (Waikato U.), Jan Van den Bussche (U. Hasselt), Margus Veanes (Microsoft Research, USA), Charles Wallace (Michigan Technological U.), Heike Wehrheim (U. Paderborn).

A special thanks goes to Prof. Hermann Maurer and the J.UCS team for the opportunity they gave us to document in another special issue of their journal some of the recent progress in the theory and applications of the ASM method, and last but not least Dana Kaiser for the efficient and pleasant cooperation.

References

[Abr96] Abrial, J.-R. *The B-Book*. Cambridge University Press, Cambridge, 1996.

[AFL08] Altenhofen M., Friesen A., and Lemcke J. ASMs in service oriented architectures. *This Journal*, 2008.

[AG08] Abrial, J.-R. and Glässer, U., editors. *Rigorous Methods for Software Construction and Analysis – Papers Dedicated to Egon Börger on the Occasion of His 60th Birthday*, volume 5115 of *LNCS*. Springer, 2008.

[AMH05] Barros, A., Dumas, M., ter Hofstede, A. Service interaction patterns. In *Proc.3rd International Conference on Business Process Management (BPM2005)*, LNCS, pages 302–318, Nancy, 2005. Springer.

[BB05] Barros, A., Börger, E. A compositional framework for service interaction patterns and communication flows. In K.-K. Lau and R. Banach, editors, *Formal Methods and Software Engineering. Proc. 7th International Conference on Formal Engineering Methods (ICFEM 2005)*, volume 3785 of *LNCS*, pages 5–35. Springer, 2005.

[BB08] Batory, D., Börger, E. Modularizing theorems for software product lines: The Jbook case study. *This Journal*, 2008.

[BBBB08] Börger, E., Bowen, J., Butler, M., Boca, P., editors. *Abstract State Machines, B and Z*, volume 5238 of *LNCS*. Springer-Verlag, 2008.

[Bel08] Bella, G. What is correctness of security protocols? *This Journal*, 2008.

[BG07] Andreas Blass, A., Gurevich, Y. Ordinary interactive small-step algorithms, I-III. *ACM Transactions on Computation Logic*, 7/8, 2006/07.

[BGR03] Börger, E., Gargantini, A., Riccobene, E., editors. *Abstract State Machines 2003–Advances in Theory and Applications*, volume 2589 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[BGRR06] Blass, A., Gurevich, Y., Rosenzweig, D., Rossman, B. Interactive small-step algorithms I-II. Technical Report 2006-170/1, Microsoft Research Redmond, November 2006. To appear in: Logical Methods in Computer Science.

[Bör02] Börger, E. The origins and the development of the ASM method for high-level system design and analysis. *J. Universal Computer Science*, 8(1):2–74, 2002.

[Bör07] Börger, E. Construction and analysis of ground models and their refinements as a foundation for validating computer based systems. *Formal Aspects of Computing*, 19:225–241, 2007.

[Bör08a] Börger, E. The Abstract State Machines method for high-level system design and analysis. In P. Boca, editor, *BCS-FACS Seminar Series Book*. 2008.

[Bör08b] Börger, E. Linking the meaning of programs to what the compiler can verify. In Meyer, B. and Woodcock, J., editors, *Verified Software: Theories, Tools, Experiments*, volume 4171 of *LNCS*, pages 325–336. Springer, 2008. First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10-13, 2005, Revised Selected Papers and Discussions.

[BR97] Bella, G., Riccobene, E. Formal analysis of the Kerberos authentication system. *J. Universal Computer Science*, 3(12):1337–1381, 1997.

[BRS00] Börger, E., Riccobene, E., Schmid, J. Capturing requirements by Abstract State Machines: The light control case study. *J. Universal Computer Science*, 6(7):597–620, 2000.

[BS03] Börger, E. and Stärk, R.F. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.

[BS07] Börger, E. and Slissenko, A. Special asm issue of fundamenta informaticae. Fundamenta Informaticae, 2007. Volume 77 (issues 1-2) with Selected Revised Papers from ASM'05.

[BT08] Börger, E. and Thalheim, B. A method for verifiable and validatable business process modeling. In Börger, E. and Cisternino, A., editors, *Advances in Software Engineering*, volume 5316 of *LNCS*. Springer-Verlag, 2008.

[Cav00] Cavarra, A. *Applying Abstract State Machines to Formalize and Integrate the UML Lightweight Method*. PhD thesis, University of Catania, Sicily, Italy, 2000.

[CRS03] Cavarra, A., Riccobene, E., Scandurra, P. Integrating UML static and dynamic views and formalizing the interaction mechanism of UML state machines. In Börger, E., Gargantini, A., Riccobene, E., editors, *Abstract State Machines 2003–Advances in Theory and Applications*, volume 2589 of *Lecture Notes in Computer Science*, pages 229–243. Springer-Verlag, 2003.

[EGG+01] Eschbach, R., Gässer, U., Gotzhein, R., v. Löwis, M., Prinz, A. Formal definition of SDL-2000 – compiling and running SDL specifications as ASM models. *J. Universal Computer Science*, 7(11):1025–1050, 2001.

[F+] Farahbod, R., et al. *The CoreASM Project*. www.coreasm.org.

[FGG06] Farahbod, R., Gervasi, V., Glässer, U. CoreASM: An Extensible ASM Execution Engine. *Fundamenta Informaticae XXI*, 2006.

[FGV04] Farahbod, R., Glässer, U., Vajihollahi, M. Specification and validation of the Business Process Execution Language for web services. In Zimmermann, W., and Thalheim, B., editors, *Abstract Sate Machines 2004*, volume 3052 of *Lecture Notes in Computer Science*, pages 78–94. Springer-Verlag, 2004.

[GK97] Glässer, U., Karges, R. Abstract State Machine Semantics of SDL. *J. Universal Computer Science*, 3(12):1382–1414, 1997.

[GRS08] Gargantini, A., Riccobene, E., Scandurra, P. A metamodel-based language and a simulation engine for Abstract State Machines. *This Journal*, 2008.

[Kru03] Kruchten, P. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 3rd edition, 2003. ISBN-10: 0321197704, ISBN-13: 978-0321197702.

[MW08] Meyer, B., Woodcock, J., editors. *Verified Software: Theories, Tools, Experiments*, volume 4171 of *LNCS*. Springer, 2008. First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10-13, 2005, Revised Selected Papers and Discussions.

[Obe03] Ober, I. An ASM semantics for UML derived from the meta-model and incorporating actions. In Börger, E., Gargantini, A., Riccobene, E., editors, *Abstract State Machines 2003–Advances in Theory and Applications*, volume 2589 of *Lecture Notes in Computer Science*, pages 356–371. Springer-Verlag, 2003.

[OL08] Ouimet, M., Lundqvist, K. The timed Abstract State Machine language: Abstract Sstate Machines for real-time system engineering. *This Journal*, 2008.

[RFV06] Glässer, U., Farahbod, R., Vajihollahi, M. An Abstract Machine Architecture for Web Service Based Business Process Management. *International Journal on Business Process Integration and Management*, 1(4):279–291, 2006.

[Sch08] Schellhorn, G. ASM refinement preserving invariants. *This Journal*, 2008.

[SV08] Slissenko, A., Vasilyev, P. Simulation of timed Abstract State Machines with predicate logic model-checking. *This Journal*, 2008.