Guest Editorial: Special Issue on Multi-paradigm Modeling

Juan de Lara  
Universidad Autónoma de Madrid,  
Spain  
Juan.deLara@uam.es

Tihamér Levendovszky  
Budapest University of Technology and Economics,  
Hungary  
tihamer@aut.bme.hu

Pieter J. Mosterman  
The MathWorks, Natick, MA,  
USA  
pieter.mosterman@mathworks.com

For decades, computational modeling and simulation have been valuable tools in tackling specific problems in the design of engineered systems. More recently, however, computational methods have moved from being applied to solve specific problems such as algorithm design, to becoming essential at the system design level. As an exponent of this trend, Model-Based Design has become key to being successful and competitive in the design of computer-based systems. Model-Based Design promotes computational models to prime deliverables in design, which has allowed inter- and intra-departmental communication based on a shared representation. This has been critical in minimizing design misunderstandings, eliminating reimplementation errors, and reducing design iterations.

The autocatalytic effect of using models at the system level and across organizations is putting forward an increasing set of demanding requirements on the theory, methodology, and technology of modeling and simulation. Modeling and simulation are indispensable for activities such as analysis, synthesis, design, understanding, testing, and optimization of systems, yet the complexity of modern systems is rapidly increasing. Though difficult to characterize, these systems share their heterogeneity of components, large number of individual interacting components, large size of components, intricate component dependency, and even non-modular interactions. Consequently, there is a commensurate intrinsic complexity in the modeling and simulation of complex systems that must be solved or resolved.

Specific requirements derive from the need to serve across design stages, not only throughout one organization but also between organizations. As such the system under design requires representations in different formalisms and with different abstractions. These representations must be coupled, combined, integrated, and transformed. Computer Automated Multi-Paradigm Modeling (CAMPaM) attempts to address these needs by integrating three research dimensions: (i) multi-formalism modeling, (ii) modeling using multiple abstraction levels, and (iii) modeling of the modeling formalisms, where the modeling of formalisms that relate and transform models takes a prominent place. In this way, CAMPaM advocates the use of the most appropriate formalism, with the most appropriate abstraction, to describe each part of the system. This helps reduce the accidental complexity because of an inappropriate or too low-level notation, and fosters the use of Domain-Specific (visual) Languages (DSLs) for modeling. Hence, CAMPaM recognizes the need for defining both the syntax and semantics of modeling DSLs, so as to customize them for specific purposes and needs. Given the requirements for Model-Based Design, these DSLs then must be interoperable and transformable.

To corroborate these observations and to underscore the timeliness of this special issue, it is noted that the
The Special Issue contains a selection of extended and were submitted, the Special Issue comprises two issues. To facilitate the number of quality papers that related fields. To facilitate the number of quality papers that were submitted, the Special Issue comprises two issues. The Special Issue contains a selection of extended and improved papers from the Second Workshop on Multi-Paradigm Modeling: Concepts and Tools—held as a satellite event of the MODELS’07 conference—as well as additional contributions from researchers that were unable to participate in the workshop. This first issue contains five carefully selected papers that cover a broad range of research challenges in Multi-Paradigm Modeling:

- In “Exploring Multi-Paradigm Modeling Techniques,” C. Hardebolle and F. Boulanger present a survey of CAMPaM techniques, especially concentrating on the problem of multi-formalism modeling. The authors first identify the sources of heterogeneity, and then go on to show some multi-paradigm techniques to address the heterogeneity of languages, the management of heterogeneous components (based on transformation, composition of languages or models, and co-simulation or unifying semantics), and heterogeneous interactions. The article concludes by presenting some desired quality aspects of the techniques regarding the support for both an open set of modeling languages and for verification methods.

- In “Patterns for Automatic Generation of Models for Soft Real-Time Systems,” J. Voeten, O. Florescu, B. Theelen, and K. Corporaal present a formal, model-based approach to analyze the behavior of soft real-time systems, such as streaming applications. In contrast to hard real-time systems, these applications allow occasional schedule overruns. Worst-case timing analysis is then inappropriate as it results in over-constrained, and thus overly expensive, systems. Instead of a worst-case analysis, the authors propose the analysis of the expected deadline miss ratio, based on the formal modeling language POOSL (Parallel Object-Oriented Specification Language). To exploit knowledge of previous designs, the authors give a number of parameterized modeling patterns that describe typical components of streaming applications such as periodic and aperiodic tasks, schedulers, resources, and event generators. Then, by merit of using a pattern-based description language, patterns can be instantiated into configurations that can be translated into POOSL for the analysis of performance properties by means of simulation.

- In “Composable Cellular Automata,” G. Mayer and H. Sarjoughian address the issue of coupling Cellular Automata models with other (non-cellular) formalisms, thus dealing with the multi-formalism concern in CAMPaM. In this manner, the authors present a system-theoretic formulation of Cellular Automata (which includes a representation of inputs and outputs, as well as timed specifications) inspired by DEVS. Furthermore, the work establishes a poly-formalism approach to model composition that relies on an intermediate interaction model that mediates between the composed models.

- In “eUDEVS: Executable UML with DEVS Theory of Modeling and Simulation,” J. Risco-Martín, S. Mittal, B. Zeigler, and J. Cruz present an approach for integrating the Unified Modeling Language (UML) and Discrete-Event System Specification (DEVS), thus bridging MDSE and modeling and simulation techniques and technology. In particular, the authors define transformations back and forth between one of several UML diagrams—representing both structure and behavior—and DEVS. As a result, multi-formalism is used for modeling by allowing the use of different UML diagrams (component, class, and package diagrams for structure; sequence, timing, and state machines for behavior), while DEVS is used to provide a formal basis for execution and analysis.

- In “Formal Specification and Analysis of Domain Specific Languages using Maude,” J. Rivera, F. Durán, and A. Vallecillo propose the use of the rewriting logic system Maude for the specification of both the syntax and dynamic behavior of modeling languages, a core research topic in CAMPaM. In particular, metamodels are encoded as Maude specifications, models are represented as terms, and the behavior is modeled by declarative rewrite rules. From this, in addition to reasoning about the models, the formal nature of Maude allows performing model simulation, as well as formal analyses such reachability analysis and linear temporal logic model checking.

- In “Engineering the Dynamic Behavior of Meta-modeled Languages,” T. Mészáros, G. Mezei, and H. Charaf present a model-driven approach to the specification of animations for DSLs, based on (timed) events. As such, the authors address the
important CAMPaM area of language engineering. The work proposes the use of three different DSLs: one to define event handlers, another one for the animation itself (based on state machines), and the third one for the configuration of the user interface. The code that is synthesized from these models is executed by a DEVS simulator, which is the engine necessary to perform the animation.

We hope that the selected papers in the two issues of the Special Issue provide the reader with a valuable overview of the CAMPaM research area, both from a theoretical as well as a practical perspective. We certainly wholeheartedly endorse the quality of the work presented and we would like to sincerely thank the authors for their contributions. Our thanks also go out to those whose submission did not make it in, given the competitive nature of this Special Issue. In that light, all of the hard volunteer work by the referees is gratefully acknowledged. Their unbiased and careful review efforts have been essential to the level of quality of the Special Issue. This is as much your achievement as it is ours. Finally, a special word of thanks goes out to Vicki Pate, our Managing Editor. Her help has been invaluable and made the entire undertaking as effortless and enjoyable as possible. Juan de Lara was partially sponsored by the Spanish Ministry of Science and Innovation, project METEORIC (TIN2008-02081/TIN).