Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

ARE SIMULATION STANDARDS IN OUR FUTURE?

Hans Ehm
Infineon Technologies AG
Am Campeon 1-12, 85579 Neubiberg, Germany

Leon McGinnis
Sch. of Industrial & Systems Engineer-
Georgia Institute of Technology
Atlanta, GA 3032-0205, USA

Oliver Rose
Ins. of Applied Computer Science
Dresden University of Technology
Dresden, 01062, GERMANY

ABSTRACT

This panel seeks to initiate a discussion within the production system simulation community about a fundamental change in the way we think about, teach, and implement production system simulation. Today, production system simulation, while based on formal simulation languages, is largely an *artistic* process. We teach production system simulation as a *studio* course, i.e., students learn an *esthetic* for production system simulation, learn by example, and their progress is evaluated through *studies* in which they create simulations. We are not surprised, in fact fully expect that two simulationists will create possibly quite different simulations of the same production system, even using the same language.

1 INTRODUCTION

"Simulation modeling" is a popular topic—a Google search returns over three million hits; if the search is restricted to 'production system" simulation modeling,' Google still returns over 65,000 hits. Perhaps this is a testament to the difficulty of doing and managing the process of creating discrete event simulation models of complex production systems.

Textbooks, monographs, and journal articles on manufacturing simulation modeling often address the process, proposing or demonstrating a workflow. A typical suggestion is the following, given by (Nubile, Ambrose and Mackarel 2004):

Table: Modeling and simulation workflow

1. **Problem Definition.** Clearly defining the goals of the study so that we know the purpose,
2. **Project Planning.** Being sure that we have sufficient and appropriate resources to do the job.
3. **System Definition.** Determining the boundaries and restrictions
4. **Conceptual Model Formulation.** Developing a preliminary model either graphically or in pseudo-code to define the components, and interactions (logic) that constitute the system.
5. **Preliminary Experimental Design.** Selecting the factors to be varied, and the
6. **Input Data preparation.** Identifying and collecting the input data
7. **Model Translation.** Formulating the model in an appropriate simulation language
8. **Verification.** Does the simulation correctly represent the data inputs and outputs?
9. **Validation.** Can the model be substituted for the real system for the purposes of experimentation?
10. **Final Experimental Design**. Designing an experiment that will yield the desired information
11. **Experimentation**. Executing the simulation to generate the desired data and sensitivity analysis.
12. **Analysis and Interpretation**. Drawing inferences from the data generated by the simulation runs.
13. **Implementation and Documentation**. Reporting the results, putting the results to use,

The focus of this panel is steps 3, 4, and 7, the steps that create the computational representation of the production system being analyzed. This is not to say the other steps are unimportant, but only to bound the context for a discussion of production system simulation standards.

Steps 3 and 4 in Table 1 create the “descriptive model” of the production system. If this descriptive model is inadequate in any respect—it fails to describe important features or behaviors, or it describes them incorrectly—then no amount of subsequent data collection, analysis, coding, testing, or computing can correct this deficiency. The likely outcome is analysis results that are not as valuable as expected or required.

Step 7 is the creation of the computational implementation of the conceptual model, most often using a COTS simulation language and associated solver or engine. This requires not only a solid working knowledge of the COTS tool(s) being used, but also the capability to understand the “descriptive model” from steps 3 and 4, and correctly translate it to the COTS tool implementation.

In reality, steps 3, 4 and 7 are not a simple linear process; considerable iteration will take place as the coding progresses and ambiguity is discovered in the system definition or the system conceptual model. In fact, the absence of formal tools for creating the system description or system conceptual model virtually guarantees ambiguity. It should be noted that this is probably not a defect in the process—an attempt to “fully” document the system description and conceptual model prior to computational implementation would likely lead to the creation of far more information than is actually needed, and perhaps to description and conceptual model documents that are so unwieldy that they are ignored (or at least avoided) by the analysts doing the coding.

Suppose we stipulate that the information needed for model translation is available, perhaps through a very fast iterative process involving customers, managers, and analysts. The task of model translation, or simulation program coding, remains a significant challenge. This is true whether one is using a simulation language at a high level of abstraction, such as ProModel™ (<www.promodel.com>, or a language that enables very low level coding of attributes or behavior, such as AnyLogic™ (<www.xjtek.com>). For the high level language, the constructs in the language may not exactly match a structure or behavior described in the conceptual model, requiring creative “work-arounds”. In the low level language, it will be necessary to “invent” code level representations of production system structure or behavior described in the conceptual model, where the information content of the coded model is significantly greater than the information content of the conceptual model on which it is based. Clearly, there will be alternative ways to create this additional information content, thus opportunities to introduce errors in the coded representation.

2 STANDARDS

According to Wikipedia, “A standard is a document that establishes uniform engineering or technical specifications, criteria, methods, processes, or practices.” Because a standard defines a “uniform” solution, it necessarily reduces the space of feasible solutions for problems where it applies. So why would we want standards? There are several possible reasons.

Standards reduce the cost of “reinventing the wheel.” If somebody already has invented a good solution to our problem, and it is economical to use that solution, why shouldn’t we? Standards can capture “best practice” in a form that makes it easily transferable and readily reusable, thus allowing new practitioners to benefit from the experiences of experts. Using standards can reduce the time and cost of development and improve the quality of result.

Standards promote “interoperability.” All our electrical appliances connect to the electric power grid through standard interfaces. As a result, somebody who invents a better appliance can deploy it to the marketplace quite easily. Communication and software standards have similar impacts on interoperability of solutions.

A number of studies have shown that standards promote market development and growth. While the direct cause is not clear, it has been argued that one reason for this effect is the reduction in uncertainty about products which are standards based.

This is not a brief in favor of standards, which carry with them a number of issues, such as the cost of development, promulgation, and maintenance. Rather it is an attempt to start a discussion about the role of standards in the future of manufacturing simulation.

3 A FRAMEWORK

A framework for discussing the issue of standards in production system simulation is suggested in Figure 1 below. The simulation code development process consumes a conceptual model and produces a simulation code. It uses a simulation language (which might be a general purpose programming language such as C), and may also use templates or libraries of code fragments, standard functions, etc.

Figure 1 suggests four opportunities for standards in production system simulation. The most obvious opportunity is the development of standard templates and libraries of standard modules. While it is true that templates and module libraries already are a common feature of many simulation languages, these are by no means standards. In the first place, there is little if any commonality across different simulation languages. Moreover, in many cases the details of the resulting computations are not known to the user, so it is difficult or impossible in complex models to judge if the resulting simulation does, in fact, accurately portray the conceptual model without extensive numerical tests. A difficulty with standards for templates and libraries is that making them generic (i.e., applicable across the entire manufacturing domain) may be ineffective—such generic templates or libraries may not adequately capture the details of a particular domain (e.g., global supply chains, semiconductor manufacturing, or robotic assembly).

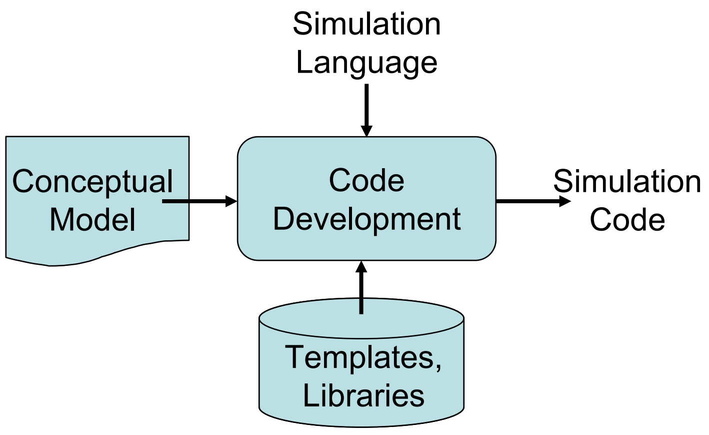

Figure 1: Simulation Code Development Process

A second opportunity for standards in production system simulation is the specification of the conceptual model. A practical, economical, uniform approach to creating and articulating the conceptual model would go a long way toward reducing the effort and the potential for error in the code development step. To be practical, the specification standard would have to be accessible to both those creating the conceptual model and those using the conceptual model to do code development, which means there would have to be convenient tools for authoring a compliant specification, and for viewing and manipulating the specification.

A recent development in software engineering is “model driven architecture,.” Or MDA, which, according to Wikipedia, “provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems.” The key to MDA is a formal approach to modeling, which in the software engineering world usually means UML, or the “unified modeling language.”

Some progress in specification standards for production system simulation has been made by NIST, drawing on the methods and tools becoming common in MDA; see, e.g., (Johansson *et al* 2008) and (Lee *et al* 2008) for descriptions of some tests and prototype implementations of a Core Manufacturing Simulation Data specification, or CMSD. CMSD specifies the data necessary to describe a factory, including the resources in the factory, the products being produced, and the operations necessary for production. The draft CMSD standard runs to 162 pages and requires some familiarity with UML for understanding; it can be accessed at <http://discussions.sisostds.org/file.asp?file=CMSDBallotingDraft-ReviewVersion.pdf>.

What CMSD lacks is an authoring tool for creating the content of the specification. The Object Management Group (OMG) has recently published a standard for a new graphical “systems modeling language,” or SysML (the specification and much tutorial information can be seen at < http://www.omgsysml.org/>). Because SysML is implemented as a profile of UML, almost every commercial UML tool also supports SysML. SysML is a natural choice as a system description tool, and there have been some initial efforts in that direction (Huang *et al* 2007, Huang *et al* 2008).

The third opportunity for standards is in the code development step itself, i.e., in the translation from a conceptual model to a simulation code. Some early attempts at this were reported over twenty years ago—see, e.g., (Haddock 1988)—but there has been little practical impact from this approach. There are good reasons to take a fresh look at this idea. Part of the MDA movement is “model transformation,” which, according to Wikipedia, “takes as input a model conforming to a given metamodel and produces as output another model conforming to a given metamodel.” Model transformation tools, such as Viatra (Csertán *et al* 2002) and MOFLON (Amelunxen *et al* 2008) are beginning to appear.

The fourth opportunity is for standards in the simulation languages themselves. Such standards would impact the semantics of simulation modeling rather than the statistical or computational aspects of languages. Looking at today’s production system simulation marketplace, it is difficult to imagine widespread adoption of such standards. However, the situation could well change significantly in the future.

4 A VIEW FROM PRACTICE

“The global supply chain is our new Fab!

Diminishing trade barriers makes global resource sharing attractive. Enhanced information technologies make it possible (manage and controllable). Today “best fitting needs” global supply chains are a must for survival in the IT, electronics and semiconductor business.

Consider production of a highly integrated chip from a platform chipset for mobile communication. It may initially be manufactured (manufactured not prototyped!) using the only technically capable supply chain, e.g.

- Wafer fabrication in Germany,
- Bumping in Taiwan (maybe the only option),
- Testing back in Germany (close link to Wafer fab),
- Assembly in Korea,
- Final Test back in Germany.

The question for simulation and modeling is: “What chain(s) are most feasible to achieve economic production at minimum risk?” recognizing that neither steep ramps nor sudden failures in launching the product can be predicted well (decreasing Forecast Accuracy in a faster moving world!). The response “This supply chain can’t be modeled?” is too simple an answer and as false as it was 15 years ago for a single semiconductor Fab. The kind of advice we often hear in gatherings like the Winter Simulation conference is ”Set up object oriented models and architectures, develop smart algorithm and simulations with the best fit goal functions in your solver engines.” This response also is inadequate, because it begs the question, “How?”

When viewing statements from the 2008 MASM conference in Miami, it seems nothing has changed in the research community. Nothing has changed from the users’ need point of view. However, in the direction of implementation the barriers are today clearer than a year ago.

It seems to be recognized now more and more that standards in wording and syntax are required to reach the goal of sharing simulation results across the supply chain. Companies and also Organizations (like the SCC – supply chain council) recognize the need for standards.

How much we already are moving into “simple” standards can be seen in our daily work. As examples, we are moving towards using Wikipedia, using the Microsoft Word spell and grammar checker and maybe even a company identical glossary for the company proprietary wording. The benefits in using these “simple” standards are usually recognized immediately in the environment - meeting minutes and discussions become clearer and more focused. But also we as practitioners recognize that even global Semiconductor Wikipedia is far from the standard which we need for simulations.

We need conceptual models which are reliable, widely shared and reflect the complexity of material, information and value flow in our production systems. Those conceptual models should serve as a base for simulation codes.

Besides the very welcome standardization activities from OMG we need a “simpler” way to spread it into our Semiconductor supply chain community wording and understanding. Maybe SysML can be this bridge.

We have some experience with academic work on a UML based conceptual model – visualized in html via Rational Rose. The first issue we needed to solve was that we did not find a commonly shared top down conceptual model. To move forward we had to build one by ourselves – see Figure 2. All the complex details (e.g. product structure serving marketing, development and manufacturing need or customer structure) then had to follow this top down picture. Another challenges was to spread this model over the whole company. For that it needs to be visualized easily and it needs to become a day to day practice.

For us practitioners the level of abstraction used in Object oriented Modeling is still too far from our wording and grammar and the level of visualization simplicity is too limited for the day to day use which is needed to get the breakthrough.

However, it is increasingly well recognized that this is the way to go to master the ever increasing complexity in global semiconductor manufacturing which is not only going to even higher integration along Moore’s Law and more than Moore, but is also going to the global supply chain – being our new Fab

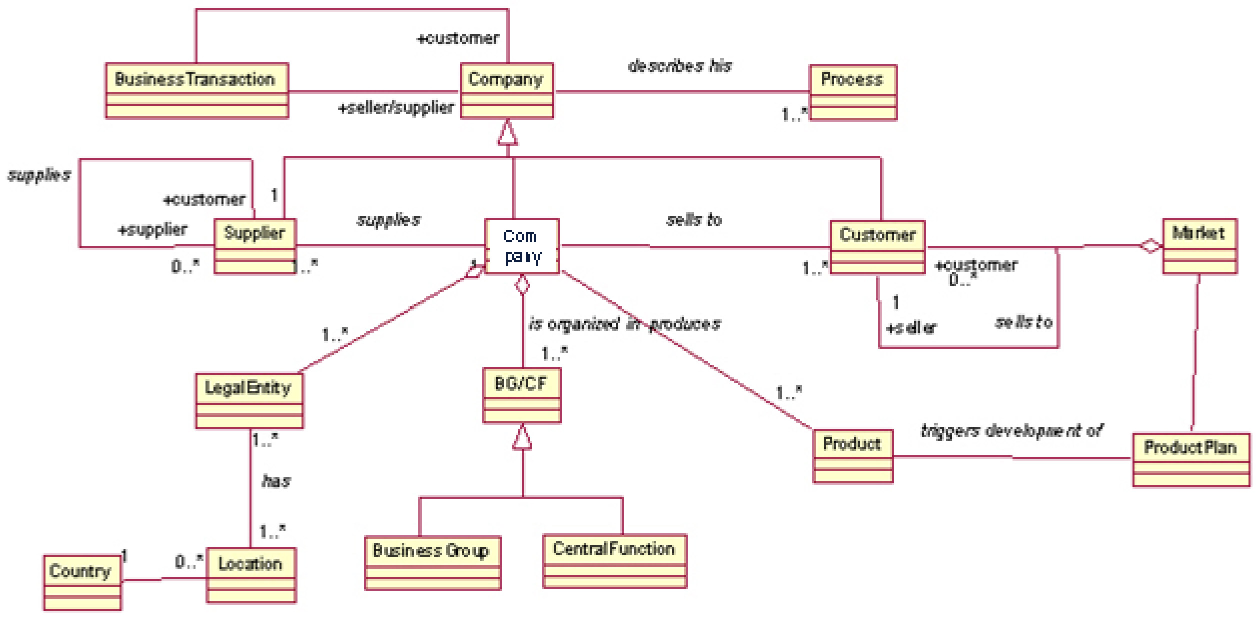

Figure 2: A a top down framework for standards

5 A VIEW FROM RESEARCH

If we intend to convince the simulation community that standards and modeling fit together nicely, we will have to provide the appropriate tools for engineers working with simulation to use the standards. It is important to note that most of these simulation experts will be domain experts and not necessarily experts in abstract modeling languages. Taking into consideration the perspective of a potential future user of standardized simulation model description approach, a number of research topics arise instantaneously.

- Current modeling tools for SysML (like MagicDraw or TOPCASED) are far too abstract for most engineers who are used to ready-to-use domain-specific simulation packages where they are supported to identify the model components they need as quickly as possible. SysML tools, however, are tools for computer scientist who tend to have a rather abstract view of the world, mainly talking about objects and classes instead of machines, jobs or processes. As a consequence, one research area will be how to customize abstract modeling tools in a way that they can be used successfully by domain experts We need to keep the abstract nature as much as possible while at the same time supporting the model developer as much as possible. Using the specification of a domain specific metamodel he should be able to build simulation models with at least the same level of support as with current simulation packages
- This leads directly to next area of research: the domain-specific meta model. We will have to develop these metamodels in order to be able to describe the base system, the control system, and the planning system both with respect to structure (i.e., its components and their relationships) and to behavior (i.e., its control rules). Current publications deal mainly with the structure and the behavior of the base system. It is an open research question how to describe the control system or even the planning system.
- After the development of the appropriate meta model, we will have to find out how the development of large model instances can be supported. There is an indication that it will be enough to store the instance data in a data base and to use the metamodel to create the full model but it is not obvious how to develop such models at a scale which is typical in industry.
- Finally, we need to develop a variety of model transformation solutions to convert the SysML model to models which can be run by current simulation packages. Research is needed to support this model transformation process. The intention is to create a general conversion tool which only needs a particular configuration file for a specific target system.

Figure 3 below illustrates one possible architecture for a process that would translate a SysML model of a problem—based on domain specific standards—to a simulation code. Figure 4 illustrates a particular realization of this process. These figures are taken from (Schönherr and Rose 2009), and based on a large research effort at Dresden University of Technology.

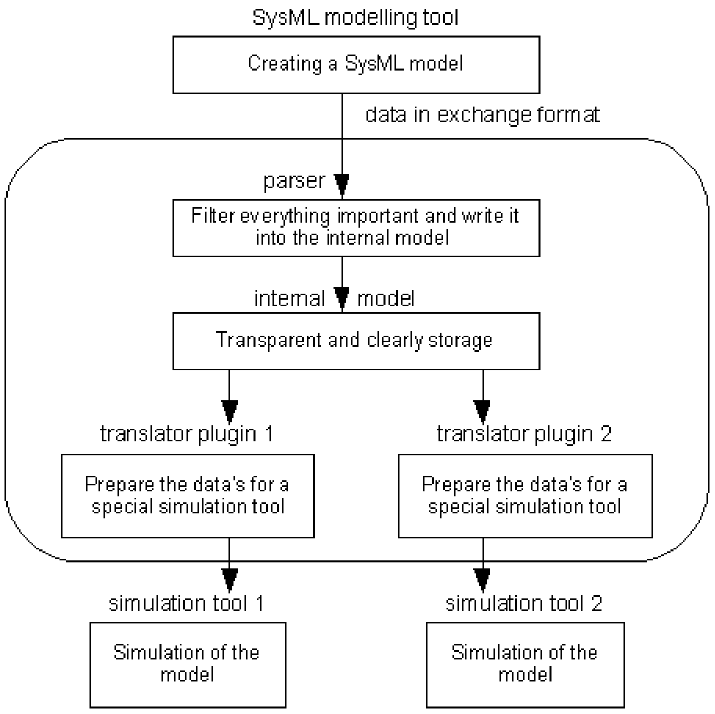

Figure 3: One architecture for translating standard SysML models to simulation

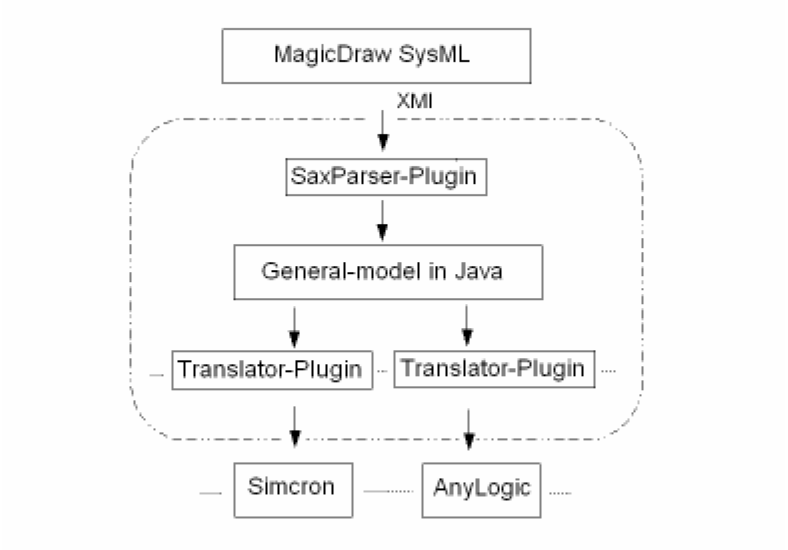

Figure 4: A specific realization of the architecture

The work at Dresden University of Technology has made significant progress; prototypes are implemented and up-and-running for the following functions:

- SysML metamodel for production systems
- SysML metamodel for assembly lines + import-plugin for our workforce scheduling system
- Converter to Anylogic, converter to Simcron Modeler
- Development of stereotypes to support the structural modeling of production systems
- Customization of TOPCASED SysML-editor-plugin

The research has involved collaboration with mechanical engineers to test our implementations. The main development environment for this work is Eclipse and Java, and the project currently engages 3 PhD students and 3 masters students.

6 A VIEW OF THE FUTURE

There is a recognized need, if not for formal standards, then at least for a more common syntax and semantics for describing production systems. There is evidence from other domains (electric/electronic circuits, fluid power systems, structures, etc) that common representation symbols and language leads to common modeling abstractions. There is a powerful new graphical systems modeling language that is being exploited by a number of users and researchers in the production systems domain. All the ingredients are there for a rapid and radical change in the technology, methodology, and practice of production system simulation. What might the future look like and what will we need to do in order to realize it?

In 2014 production system domain experts will use a SysML-based tool with a specific production system profile (meta-model) that will present appropriate modeling symbols with a syntax specific to production systems. Using this production system specific graphical modeling language, the domain expert will create a model of the problem to be analyzed, including a specification of the control processes and planning processes, and the resulting model will be such that it can be explained to the customer (problem owner) for direct validation. Model transformation technology will allow the resulting graphical system model to be converted automatically into the form needed for the customer's current discrete event simulation software.

These capabilities will be available in 2014 because in 2010 the production system simulation community decided that the time was right to begin developing a consensus production system meta-model and corresponding symbology. At the same time, forward-looking vendors of discrete event simulation software recognized the value of publishing meta-models for their simulation languages. Together, these two developments enabled the development of model transformation technology specific to the production systems and production systems simulation domains. As research prototypes demonstrated the technical feasibility of the approach, third party providers sprang up to provide customized toolsets and consulting services which fueled rapid adoption across the production systems domain. As a result, the sales of simulation software for use in production systems grew by an order of magnitude from 2009 to 2014.

7 CHALLENGE

The discussion of standards in production system simulation should address several fundamental questions, including:

- Can we make the case that the development and use of production system simulation standards will improve the practice of production system simulation, and increase the penetration of production system simulation in practice?
- What incentives and processes will be needed to encourage researchers and practitioners to develop and refine domain specific languages for creating conceptual models of production systems?
- What incentives and processes will be needed to encourage researchers and practitioners to develop standard methods for translating conceptual models into simulation code?
- How can COTS simulation tool vendors remain independent and successful if standards emerge for the semantics of manufacturing simulation?

Up to now, the discussion of standards in simulation has engaged a very small fraction of the simulation research and application community. The time is right for the discussion to move into the mainstream. The potential benefit is the emergence of new capabilities for production system simulation, and sustainable growth for the production system simulation market.

REFERENCES

Amelunxen, C., F. Klar, A. Königs, T. Rötschke, and A. Schürr. 2008. Metamodel-based tool integration with moflon. In *Proceedings of the 30th international Conference on Software Engineering.* pp 807-810. Leipzig, Germany. ACM, New York, NY.

Csertán, G. G. Huszerl, I. Majzik, Z. Pap, A. Pataricza, and D. Varró. 2002. VIATRA; Visual Automated Transformations for Formal Verification and Validation of UML Models. *Automated Software Engineering, International Conference on*, pp. 267, 17th IEEE International Conference on Automated Software Engineering (ASE'02.

Haddock, Jorge. 1988. A Simulation Generator for Flexible Manufacturing Systems Design and Control. *IIE Transactions*. 20(1):22-31.

Huang, E., R. Ramamurthy, and L. McGinnis. 2007., System and Simulation Modeling Using SysML. In *Proceedings of the 2007 Winter Simulation Conference*. eds. S. G. Henderson, B. Biller, M.-H Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 796-803. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Huang, Edward, Ky Sang Kwon, and L. F. McGinnis, "Toward On-Demand Wafer Fab Simulation using Formal Structure and Behavior Models," *Proceedings of the 2008 Winter Simulation Conference*; ed S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 2341-2349. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Johansson, M., S. Leong, Y. T. Lee, F. Riddick, G. Shao, B. Johansson, A. Skoogh, and P. Klingstam. 2007. A Test Implementation of the Core Manufacturing Simulation Data Specification, In *Proceedings of the 2007 Winter Simulation Conference*, 1673-1681. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lee, Y. T., S. Leong, F. Riddick, M. Johansson, and B. Johansson. 2007. A Pilot Implementation of the Core Manufacturing Simulation Data Information Model. In *Proceedings of the Simulation Interoperability Standards Organization 2007 Fall Simulation Interoperability Workshop*. Orlando, Florida: Simulation Interoperability Standards Organization, Inc.

Nubile, E., E. Ambrose, and A. Mackarel. 2004. Development of a procedure for manufacturing modeling and simulation. 21st International Manufacturing Conference, Limerick, Ireland.

Riddick, F. and Y. Lee. 2008. Representing layout information in the CMSD specification. *Proceedings of the 2008 Winter Simulation Conference*, ed S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 1777-1784. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schönherr, O., and O. Rose. 2009 (forthcoming). First steps towards a general SysML model for discrete process production systems. In *Proceedings of 2008 Winter Simulation Conference*, eds M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

HANS EHM is Principal of Logistics Systems of Infineon Technologies AG. He holds degrees in Physics from Germany and a M.S./OSU. In over 20 years in the Semiconductor industry he was granted managing and consulting Positions at Wafer Fabrication, at Assembly & Test and nowadays for the global Supply Chains - on production site, on business unit and on corporate level. He is Board member of camLine Holding AG an IT company for supply- and quality chains. He led Governmental supported projects on national and international level in the context of IT, Semiconductor Manufacturing and Supply Chains - within this he initiated the EU/US MIMAC co-operation on capacity modelling. He teaches Supply Chain Management at Universities. He is INFORMS member and on the International Advisory Committee of MASM. His email address is <hans.ehm@infineon.com>.

LEON MCGINNIS is the Gwaltney Professor of Manufacturing Systems at the Georgia Institute of Technology, where he serves as Associate Director of the Manufacturing Research Center, Director of the Product and Systems Lifecycle Management Center, and founder of the Keck Virtual Factory Lab. His personal research focuses on systems modeling and systems design for discrete event logistics systems, and he leads several research teams addressing large scale systems analysis problems for industry sponsors. He is a member of IEEE, INFORMS, and IIE. His email address is <leon.mcginnis@gatech.edu>.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Institute of Applied Computer Science of the Dresden University of Technology, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI. Web address: <www.simulation-dresden.com>.