Proceedings of the 2009 Winter Simulation Conference
L.H. Lee, M.E. Kuhl, J.W. Fowler, and S. Robinson, eds.

A GENERAL SYSML MODEL FOR DISCRETE PROCESSES IN PRODUCTION SYSTEMS

Oliver Schönherr
Oliver Rose

Institute of Applied Computer Science
Dresden University of Technology
Dresden, 01062, GERMANY

ABSTRACT

In many areas of science, like computer science or electrical engineering, modeling languages have been established, however, this is not the case in the field of discrete processes (Weilkiens 2006). There are two reasons which motivate such a development. First, modeling languages allow realizing projects by the principles of systems engineering. So one obtains clearness even for large projects and reduces the discrepancy between model and reality. Second, modeling languages are a central part of automatic code generation. In this paper, we present our first steps in developing a simulation-tool-independent description of production systems and first ideas on how to convert such a general model into simulation-tool-specific models.

1 INTRODUCTION

In the field of software engineering automatic code generation of UML-Models by CASE-tools is very common and standardized (Fowler 2003, p.23). For modeling discrete processes there are many approaches called "*Model Based Software Engineering*" (MBSE) like *Stateflow Coder*, ASCET, or ADAGE, but none of them has been established as a standard (Committee of Software Engineering 2004). This could be due to the lack of an adequately powerful or general modeling language. However, in particular for modeling discrete processes in production automatic code generation is useful, because there are many different tools such optimizers or schedulers which cannot exchange their non-standardized models.

The Object Management Group (OMG) developed the Systems Modeling Language (SysML) to facilitate modeling of complex systems. SysML is a standard based on the standardized general-purpose *Unified Modeling Language* (UML). There have been many disputes about SysML during the short period of time since its publication. SysML is spreading very fast, today many of the most prominent developers of modeling tools like ARTiSAN, Telelogic, I-Logix and Sparx Systems make use of SysML in their tools.

This paper presents an approach for automatic model generation of discrete processes in production. Our aim is to develop production models by means of SysML and to build converts from SysML models to a large variety of simulation tools. At first we consider whether SysML is suitable for modeling discrete processes in production. In order to understand the specifics of modeling production systems we interviewed experts, studied present literature and conducted a market analysis of simulation modeling tools. Based on this knowledge we intend to create a general possible model for discrete processes in production which permits comprehensive production scenarios. In addition we tested whether SysML is appropriate to build our general model. After the concept for building production models with SysML, we developed a practical approach for automated model generation few simulators based on SysML models.

2 MARKET ANALYSIS

To understand the specifics of modeling, we made an extensive market analysis of simulation modeling tools in combination with a survey of literature and expert interviews. Every simulation tool provides its own approach to model production scenarios since each of them attempts to build models in a comprehensive and comfortable way. We used this knowledge of modeling to understand and structure discrete processes in production.

Due to our final goal of automated model generation from given SysML models, it was also important to know the modeling peculiarities of the simulators. Furthermore we have tested whether the simulators are suitable for automated model generation (for example input/output format), in order to find a suitable tool for our prototype. A part from the ability to build models properly, we also tested other important tool properties like simulation speed.

There are many simulation tools which are applied in the field of production; we consider 24 of them. After a first examination of obvious exclusion criteria, six tools remained for our detailed analysis. We excluded all tools which are not able to model discrete processes in production, which are too slow or whose price is obviously disproportionate.

For testing the tools, we customized the criteria from the market overview "Simulationstechnik in Produktion und Logistik" to our problem (Noche and Wenzel 1991) and obtained the following list of criteria:

- modeling concept
- data import/export
- simulation speed
- statistical analysis / portability
- presentation
- costs / support costs

We defined a test scenario which we used coherently and consistently. For an efficient evaluation of our criteria the test scenario should cover a broad spectrum of modeling possibilities of production but still remain manageable. We chose a marginally modified scenario from (Law and Kelton 1991, p. 685ff.) as our test case.

In summary, Simul8, AnyLogic and Em-Plant were able to largely meet the requirements, although Em-Plant does not use the well-structured, hierarchical and, hence, for automated model generation suitable data format XML. Aditional details can be found in (Schönherr 2008) and (Bohn 2008).

3 SPECIFICATION OF A GENERAL MODEL

Due to detailed research and by making use of our market analysis, we developed a general model for discrete processes in production, which includes comprehensive production scenarios. We checked whether our model can be represented with SysML. In accordance with UML, SysML divides the model into a structural and a behavior part. The structural part describes the static structure, like the elements and their relationships, in a system. In the behavior part SysML describes the dynamic behavior from and between its elements.

3.1 The Structural Model

SysML provides four diagrams for describing the structure of a model. In 2006, Huang, Ramamurthy and McGinnis proposed how to describe the structure of a production system with SysML. In their work they use the "Block Definition Diagram" and the "Internal Block Diagram" to build a metamodel for flow shop problems (Huang et. al 2007, p. 798f). In this paper, we try to create a general model for all fields of production like flow shop or open shop problems. We also use the block definition diagram and the internal block diagram (Figure 1).

Whereas in other areas workflow is determined by information flow, in production the entity controls the behavior of the model: the entity is the central element because it represents the job or lot which moves through and is processed by the elements of the machinery. All events in a model, except for interruptions, are triggered by the entity. The entities enter the system through the arrival process and leave it through the departure process. While they travel on specified routes, different processes execute actions on them, for which the processes may use resources but they do not have to. Along their way the entities can be stored in queues.

To describe the internal relationships between the different elements (for example, the route of the entities), we use the internal block diagram like McGinnis and Huang (Huang et al. 2007, p.799). Here, we describe the different elements the entity passes through as blocks with object flow ports.

3.2 The Behavior Model

We found no existing approach for modeling the behavior in the literature. We used the knowledge that we obtained through our market analysis to define patterns for the behavior of the identified elements. We could split these patterns into phases.
The process could be split into four phases:

1. Accept the incoming entity to start the activity.
2. Attach the needed resources to start the activity.
3. Execute an action on entity.
4. Forward entity.

The arrival process consists of "create entity" and "forward entity". The departure process consists of "receive entity" and "destroy entity". The queue consist of "receive entity" , "insert entity" and "forward entity". The resource pool element only consists of "setting up".

After we split the elements into phases, we split these phases into patterns of behavior. Then we tested whether these pattern can be represented with SysML. It turned out that all patterns can be modeled with SysML activity diagrams. Details can be found in (Schönherr 2008).

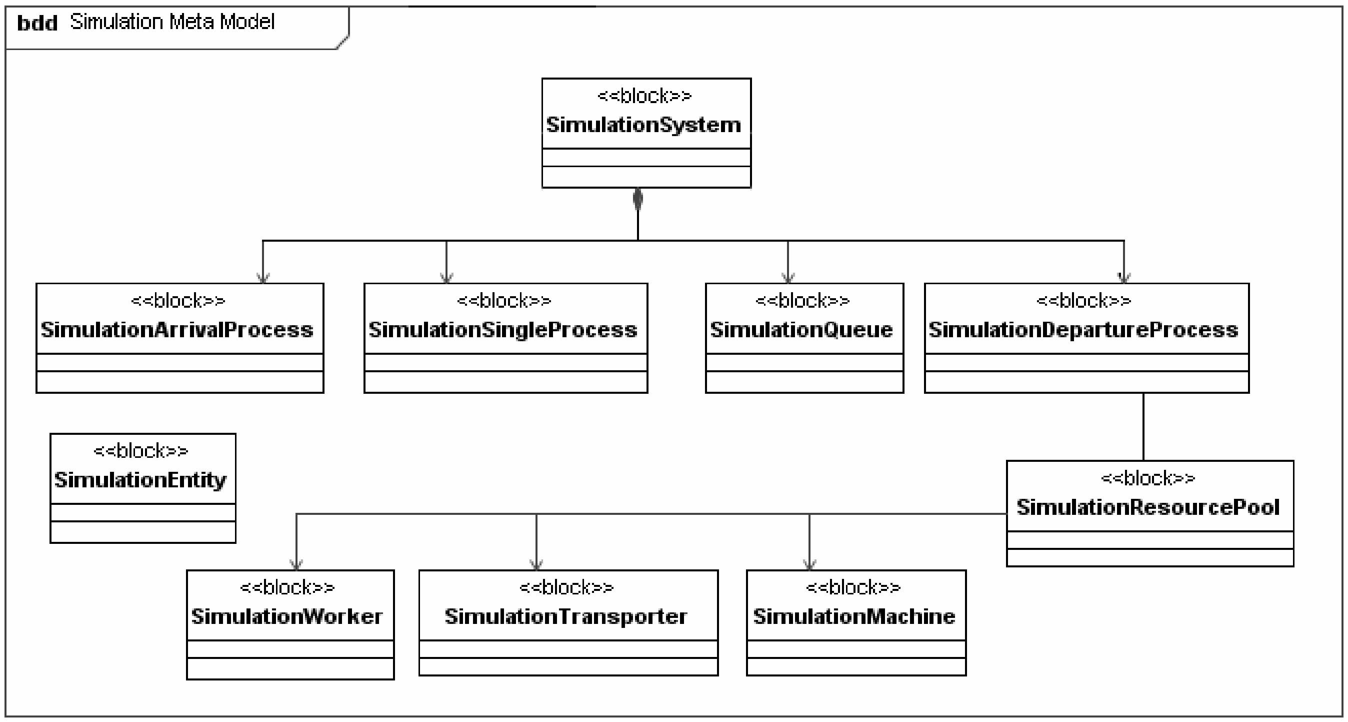

Figure 1: Structural metamodel as SysML Block Definition Diagram

4 A PRACTICAL APPROACH FOR AUTOMATED MODEL GENERATION

We developed a software that automatically generates models for simulation tools from given SysML models. In the previous section we attained that SysML has the ability to model comprehensive production scenarios. This is the basis for our approach. To build an effective tool we use a multilayer architecture (Figure 2). At first we build the model with a SysML modeling tool. The modeling tool should provide a suitable data interchange format, contain all identified SysML elements and must be appropriate for building large models.

If the SysML model is available in a suitable exchange format, it can be transformed into an equivalent of a simulation modeling tool. Since it should be possible to transform a SysML model into models of different simulation tools, a separate output must be generated for each program. Each simulation needs a suitable dedicated model in a special input format. To simplify the software architecture, the model generation is divided into two steps, which involve an additional "internal model". In the first step we used a program called "parser". The parser reads the SysML model, which is specified in the exchange format, filters out all non-relevant information, and writes the remaining significant parts into the internal model. In the second step, a program, called "translator plugin", prepares the data from the internal model for a special simulation tool. More precisely, it takes all the relevant data and translates them into the input data format of the simulation program, which is defined by rules. Since each simulator has its own format, one must write a separate translator plugin for each simulation tool.

The advantage of the proposed architecture compared to the performance of a single step conversion from a SysML file into a model for a simulation program is that the first step (the parser) does only need to be executed once. However, the architecture assigns a special role to the "internal model" because it must be particularly suited to derive models for simulation tools. The internal model must contain all information for the generation of production models and must still remain transparent.

When creating the SysML model, the model must adhere to certain structures in order to be properly recognized by the parser. These structures are defined in the metamodel. In the complete created model every object and ability must be a subset of the metamodel, otherwise the parser cannot recognize them.

Currently, we develop a prototype implementation of this architecture in JAVA including a converter to the AnyLogic simulation package.

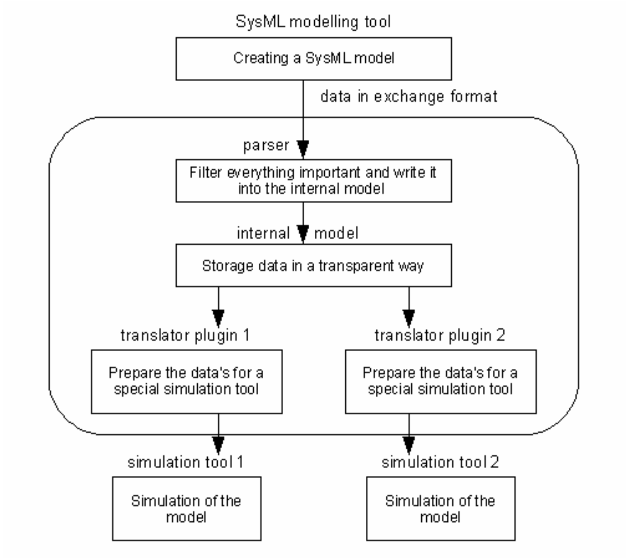

Figure 2: System architecture of the converter

5 SUMMARY AND OUTLOOK

We tried to identify and structure the significant properties of discrete processes in production systems. At the moment, we did not identify all peculiarities because the behavior of systems is very complex. We intend to extend our model as our modeling work progresses. In addition, we intend to gain new insights from working on larger sample systems from our partners in the semiconducter industry.

Furthermore, we tested whether SysML is suitable to build models based on our concept. It turned out that SysML is comprehensively usable, the structural model and all behavior patterns can be modeled with SysML.

A problem of modeling with SysML is the representation of large systems. It is obvious that modeling effort as well as clarity can be problematic. But this is a problem of all graphical modeling languages. To solve the problem, modeling must be scalable. One approach would be to separate domain model and instance model. Another possible solution would be to prepare design patterns for recurring behavior. At the moment, we are working on both solution approaches.

REFERENCES

Bohn D. 2008. Eine Marktanalyse von Simulationstools für die Modellierung von Produktionssystemen als Grundlage für die Konvertierung eines SysML-Modells in den Simcron Modeller. Unpublished internship thesis. Department of Computer Science. Dresden University of Technology.

Fachausschuss Software Engineering 2004. Code Generierung und modellbasierte Softwareentwicklung für Luft- und Raumfahrtsysteme. <http://www.t6.dglr.de/Veranstaltungen/2004_MDSE/DGLR_T64_2004_bericht.html> [31.10.2008]

Fowler M. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Pearson Education.

Huang E., R. Ramamurthy, L. McGinnis. 2007. System and simulation modeling using SysML. *Proceedings of the 2007 Winter Simulation Conference.*

Law A.M., W.D. Kelton. 2000. *Simulation Modeling and Analysis*. McGraw-Hill.

Noche B., S. Wenzel. 1991. Marktspiegel Simulationstechnik in Produktion und Logistik. TÜV Verlag Rheinland.

Schönherr O. 2008. Ein allgemeines SysML-Modell zur Abbildung diskreter Prozesse in der Produktion. Master thesis. Department of Computer Science. Dresden University of Technology.

Weilkiens T. 2006. *Systems Engineering mit SysML/UML*. Dpunkt Verlag.

AUTHOR BIOGRAPHIES

OLIVER SCHÖNHERR is a PhD student at the Dresden University of Technology. He is a member of the scientific staff at the Chair for Modeling and Simulation. He received his M.S. degree in computer science from the Dresden University of Technology, Germany. His e-mail address is <oliver.schoenherr@tu-dresden.de>

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>.