View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CiteSeerX

Basic Guidelines for Simulating SysML Models: An Experience Report

Mara Nikolaidou, George-Dimitrios Kapos, Vassilis Dalakas and Dimosthenis Anagnostopoulos
Department of Informatics and Telematics
Harokopio University of Athens
70 El. Venizelou St, Kallithea, 17671, Athens, GREECE
email: {mara, gdkapos, vdalakas, dimosthe}@hua.gr

***Abstract*—Though there are numerous efforts for simulating SysML models, the automated generation of executable simulation code for specific simulation environments without any interference by the system engineer is still an issue attracting the researchers' attention. To become efficient and easy to use, such an activity should be explored using standardized methods and tools, such as the utilization of MDA concepts for model transformation. In this paper, we identified some basic guidelines for the generation of executable simulation code based on existing SysML system models and the selection of related methods and tools for simulation and model transformation purposes. The proposed guidelines are incorporated in a three-step methodology that can be applied independently of the simulation framework selected. In the paper, we discuss our experience applying it, based on examples from different system domains, where DEVS framework was chosen for simulation purposes. The reasons for its selection and the potential drawbacks and difficulties drawn from its adoption are also discussed, to comment on the characteristics a simulation framework and language should obtain to be effectively applied for SysML model simulation.**

I. Introduction

Systems Modeling Language (SysML) is the emerging standard for model-based system engineering [1]. It is a general-purpose graphical modeling language, for systems engineering applications and supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities. It is defined as a profile of Unified Modeling Language (UML), the standard for modeling software intensive systems.

Using SysML the system engineer should perform all engineering activities based on a common model, according to Model Driven Architecture (MDA) concepts [2]. The common system model should be general enough to cover all engineering activities, while it should also be specialized to serve specific engineering activities, such as system validation.

Model-based system design is served by numerous methodologies adopting SysML as a modeling language. To validate the proposed system architectures, quantitative methods are usually applied, focusing on system performance. Thus, system validation is often performed using simulation. Since most simulation methodologies are model-based, such an approach is suitable for model-based system engineering. This justifies the increased interest in integrating SysML modeling environments and simulation tools. To this end, there are numerous efforts to simulate SysML models (for example [3], [4]). Most of them aim at transforming SysML system models to simulation models, executed in a specific simulation environment, as for example PetriNets [4] or ModelicaML [5].

Prominent efforts in this area include the definition of a SysML4Modelica profile endorsed by OMG [6] and the corresponding transformations to convert SysML system models, using the profile, to executable Modelica simulation code with standard MDA methods, as QVT language. The authors are working on a similar approach [7] targeting at transforming SysML models to executable DEVS simulation models [8]. A corresponding SysML DEVS profile and SysML-to-DEVS metamodel transformation in QVT have been implemented and tested using different system examples.

Though there are numerous efforts for simulating SysML models, automated generation of executable simulation code for specific simulation environments without any interference by the system engineer is still an open issue. To become efficient and easy to use, such an activity should be explored using standardized methods and tools, utilizing MDA concepts for model transformation. Furthermore, a variety of simulation methods and tools should be supported, depending on the systems under study. Either continuous or discrete simulation may be applied, while model libraries could be used instead of writing simulation code for all system entities. Though the simulation tools and system requirements may differ, model transformation tools and the process of integrating simulation-specific characteristics into SysML models may be standardized. Towards this direction, basic guidelines for the generation of executable simulation code based on existing SysML system models and the selection of related methods and tools for simulation and model transformation are identified. These guidelines, based on the authors experience, are analytically explained in the paper, based on specific examples. The proposed guidelines are incorporated in a three-step methodology that can be applied independently of the simulation framework selected. In the paper, we will discuss our experience applying it, based on examples from different system domains, where DEVS framework was chosen for simulation purposes. The reasons for its selection and the potential drawbacks and difficulties drawn from its adoption will also be discussed, to comment on the characteristics, a simulation framework and language should have to be effectively used for SysML model

simulation.

The paper is structured as follows: In the section II a short review of SysML model simulation is given. The proposed guidelines and the related benefits and potential considerations are discussed in section III. In section IV the three-step methodology for SysML model simulation is presented. An example applying the proposed methodology when simulating SysML models using DEVS is described in section V. Conclusions and Future work reside in section VI.

II. Related Work

There are a lot of efforts from both research and industrial communities to simulate SysML models [9], [10]. Apparently SysML supports a variety of diagrams describing system structure and states, necessary to perform simulation, which are utilized by different approaches. In most cases, SysML models defined within a modeling tool are exported in XML format and, consequently, transformed into simulator specific models and forwarded to the simulation environment.

Depending on the nature and specific characteristics of systems under study, there is a diversity of ways proposed to simulate SysML models, utilizing different diagrams. In [11], a method for simulating the behavior of continuous systems using mathematical simulation is presented, utilizing SysML parametric diagrams, which allow the description of complex mathematical equations. System models are simulated using composable objects (COBs) [12]. It should be noted that in any case SysML models should include simulation-specific information, which facilitates simulating them [13]. These approaches are better suited for systems with continuous behavior.

Simulation of discrete event systems is utilized, based on system behavior described in SysML activity, sequence or state diagrams. In [3], system models defined in SysML are translated to be simulated using Arena simulation software. SysML models are not enriched with simulation-specific properties, while emphasis is given to system structure rather than system behavior. Model Driven Architecture (MDA) concepts are applied to export SysML models from a SysML modeling tool and, consequently, transformed into Arena simulation models, which should be enriched with behavioral characteristics before becoming executable. In [4], the utilization of Colored Petri Nets is proposed to simulate SysML models. If the system behavior is described using activity and sequence diagrams in SysML, it may be consequently simulated using discrete event simulation via Petri Nets.

To enable the construction of executable simulation models, simulation capabilities should be embedded within SysML models utilizing profile mechanism. The formal method, proposed by OMG to extent or to restrict UML and consequently SysML models, is the definition of a profile, emphasizing the properties of a specific world or domain, a simulation methodology or tool in this case. SysML profiles contain stereotype definitions, which facilitate the formal extension/restriction of UML entity semantics. In [14], simulation is performed using Modelica [15]. To ensure that a complete and accurate Modelica model is constructed using SysML, a corresponding profile is proposed to enrich SysML models with Modelica-specific properties, utilizing ModelicaML.

Ideally, the simulation models extracted from SysML models should become executable without any additional programming effort from the system engineer, while SysML models should be easily transformed to executable simulation models. Both, [3] and [10] utilize MDA concepts for model transformation.

Furthermore, the simulation methodology adopted should be popular and facilitate the execution of simulation models on a variety of simulators, while the existence of model libraries may also enhance the capabilities of the system engineer to produce simulation code.

The SysML4Modelica profile endorsed by OMG [6] enable the transformation of SysML models to executable Modelica simulation code. The relevant standard also proposes the transformation of SysML models, defined using the profile, to executable Modelica models using standard MDA methods, as QVT language. The authors are working on a similar approach [7] targeting at transforming SysML models to executable DEVS simulation models [8]. A corresponding SysML DEVS profile and SysML-to-DEVS metamodel transformation in QVT have been implemented and tested using different system examples.

Since SysML profiles are based on formal UML extension mechanisms, they can be implemented in any standard UML modeling tool, such as Rational Modeler [16] or MagicDraw [17], enabling the integration of simulation tools with any of them.

In the following, we try to identify the requirements for effectively simulating SysML models, independently of the simulation method applied, and suggest some basic guidelines for selecting the methods and tools to use.

III. Guidelines

The proposed guidelines are suggested to enable engineers facing the challenge of evaluating existing SysML system models via simulation to choose the proper simulation and model transformation methods and tools. Issues concerning SysML model enrichment with simulation specific information and the generation of simulation models are addressed. Model enrichment must be based on appropriate SysML profiles, while, in order to generate simulation models, a reference meta-model for the corresponding simulation framework must be defined or selected, if one already exists. Each guideline is analyzed and justified, while open issues that may affect applicability of the guidelines are also discussed.

A. SysML profiles with simulation-specific properties

As already identified in the literature, to simulate a SysML system model the system engineer should incorporate simulation-specific properties in it. This task corresponds to the specialization of the common model to serve a specific engineering activity, in this case system validation using simulation. The appropriate way to achieve this, is by defining and

using simulation-related SysML profiles, related to a specific simulation methodology, as for example SysML4Modelica [6] or DEVS-SysML [18]. Such profiles should be applied in modeling tools (e.g, MagicDraw) and enable the validation of simulation-enriched SysML models prior to their transformation to simulation code.

Benefits:

- Definition and use of appropriate profiles allow the addition of simulation-specific properties in the common model, instead of redefining the simulation model.
- The adoption of such formal methods, offers credibility and the possibility to learn, extend, restrict and employ modeling tools, without specialized knowledge about specific simulation environments.
- Model validation is supported.
- Focus remains on analysis/design of the system.

Considerations:

- Definition of a SysML simulation-related profile is not a trivial procedure. Required simulation-specific attributes must be identified and positioned in the appropriate elements of the system model.
- Structural correlation between SysML and simulation models does not suffice for the definition of a profile. Semantics of the simulation formalism and those of SysML should also be taken into account.
- There are no SysML profiles available, for the majority of the simulation environments.
- The profiles should be applied and tested in the majority of well known modeling tools.

B. Complete enrichment of SysML models

It is more efficient to include all simulation-related information within the SysML model and automatically produce fully executable simulation code, since in this case the simulation method becomes transparent to the system engineer. There is no need to write simulation code or learn the specifics of the simulation environment. The engineer only uses SysML notation to describe the system model and its simulation-specific behavior. This feature is not supported by most of the approaches recorded in the literature, that focus on transforming system structure to the simulation model and not simulation-related system behavior.

Benefits:

- Successful incorporation of simulation behavioral (besides structural) aspects in the central model enables automated generation of executable simulation models.
- Such an approach enhances credibility and offers transparency and usability on the whole process.

Considerations:

- Although models may be validated against a profile, conceptual completeness of the profile itself in regard to the simulation framework cannot be easily proved.
- Defining simulation model behavior in a precise, yet generic manner is a challenging task.

C. Generating simulation models from system models

Transformation of SysML to simulation-specific models should be accomplished in a standardized fashion based on MDA concepts, using existing languages and tools, as QVT [2]. Such languages offer high-level mapping constructs that create simple and maintainable transformations that can be executed in several environments. To facilitate such a transformation, the existence and acceptance of a MOF meta-model for the applied simulation methodology is essential. Such a standardized meta-model for a specific simulation domain is of greater value, when the same simulation methodology can be applied using different tools, e.g., DEVS methodology.

Benefits:

- Simpler transformation definition, in comparison to custom code writing.
- Enhanced credibility, transparency and (re-)usability on the whole process.

Considerations:

- Completeness of such transformations is not always easy to check.
- Such an approach introduces overhead that is, however, overridden by the benefits, as the complexity of system models rises.

D. Reference meta-models

In the case where there is a variety of different simulators supporting the same simulation methodology, the simulation-specific MOF meta-model may serve as the basis for platform-independent models (PIMs) towards two directions: a) enable SysML to simulation-specific model transformation, and b) enhance interoperability between different simulators independently of the language they use (e.g. C++, Java) and the way they are executed (centrally or distributed). The progressive development of such MOF meta-models for different simulation environments may further contribute to interoperability and exploitation of a common SysML profile, serving all of them.

Benefits:

- The existence of reference meta-models enables QVT-based transformations.
- Interoperability between different simulation tools within a specific simulation framework is promoted.

Considerations:

- Currently, there are not many simulation-specific reference meta-models available.
- Reference meta-models are fully utilized once they have become widely accepted standards. However, this standardization may be a long-lasting procedure.

IV. A Methodology for SysML System Model Simulation

Having these issues in mind, a three-step methodology is proposed for SysML model simulation independently of the simulation framework or language adopted, as depicted in Fig. 1. The system engineer should specify the system model

using SysML via a modeling tool and receive valid simulation results from the execution of the corresponding system model in an appropriate simulation environment. As shown in the figure, the three discrete steps are:

Step 1: *Constructing a simulation-specific profile to enrich SysML system models with simulation capabilities.* The system engineer should enrich system models with simulation information. This should be performed during system modeling, within the SysML modeling tool, according to the specific profile. The simulation-specific profile must consist of both a set of stereotypes and constraints. Stereotypes are used to characterize specific SysML model elements, while constraints specify how valid simulation models should be created.

Step 2: *Transforming SysML system models (usually represented in XMI) to simulation-specific system models.* All standard modeling tools facilitate exporting SysML models in XMI format. It is considered as a standard platform-independent representation of UML/SysML models. To enable model transformation in a standardized fashion, a platform-independent MOF 2.0 meta-model for the corresponding simulation environment should be available. Provided that such a meta-model is available, OMG's Query/View/Transform (QVT) language can be used for model transformation using existing transformation tools [7].

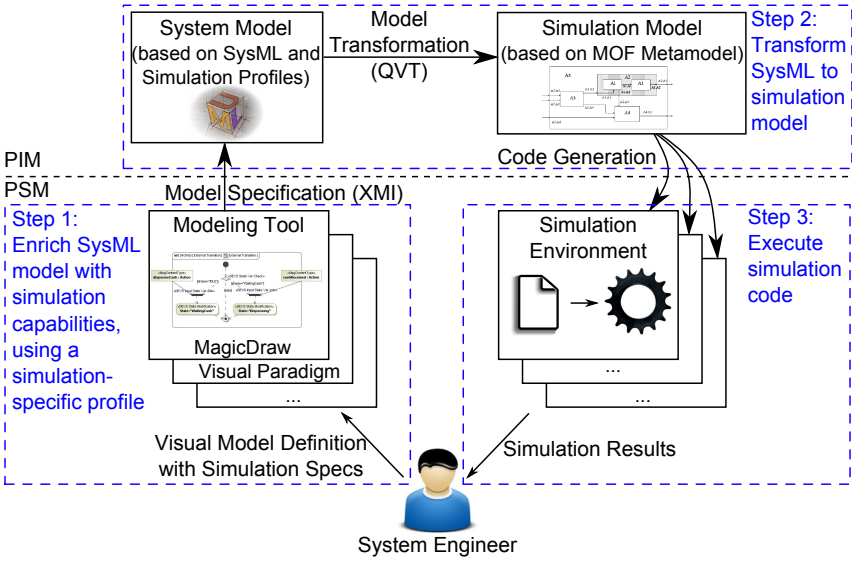

Fig. 1. A methodology for simulating SysML models

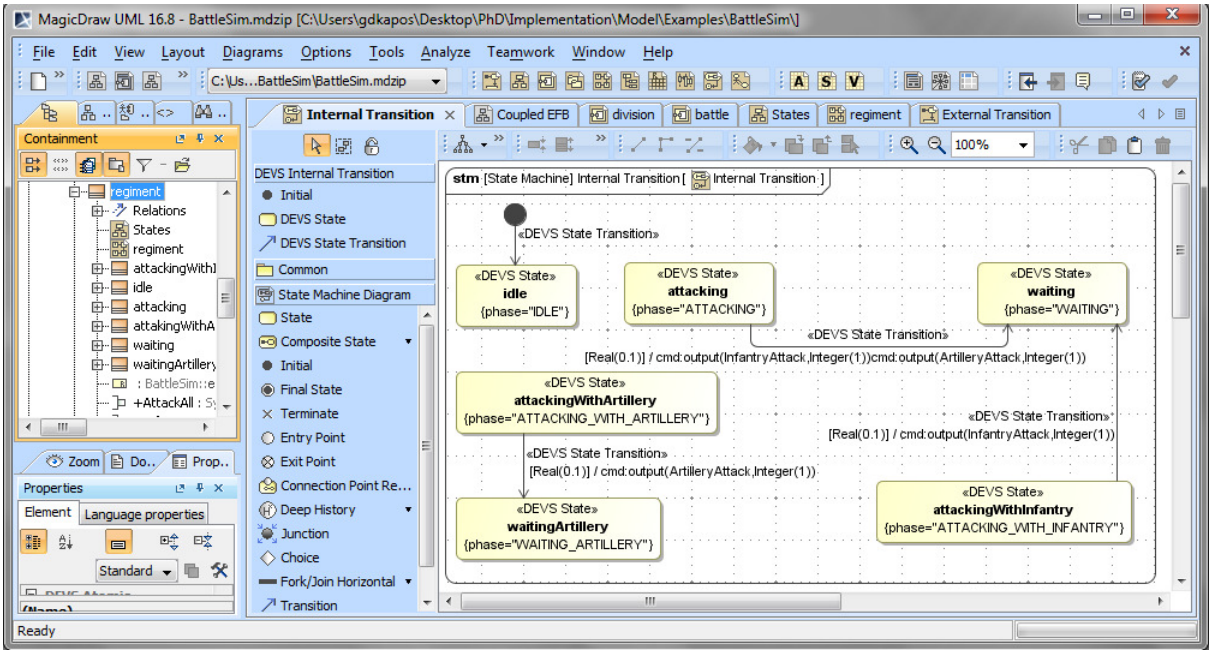

Fig. 2. Enriching SysML models using the DEVS-SysML profile in MagicDraw

Step 3: *Creating executable simulation code.* The transformation of simulation-specific MOF models to executable simulation code constitutes the last step of the proposed approach. This transformation heavily depends on the target simulation environment. We argue that the maturity of a simulation community, in providing required tools that enable such transformations and defining a MOF meta model, plays a significant role in selecting it. After all, using SysML is closely related to the utilization of existing standards and tools and simulation frameworks should move towards their integration in model-driven engineering frameworks.

V. Simulating SysML Models with DEVS

Based on the presented methodology, the automated DEVS executable simulation code generation from enriched SysML models has been realized [7]. DEVS was selected as the target simulation framework for a series of reasons. DEVS is a formalism allowing a hierarchical and modular description of the models, which lacks a standardized and easy-to-use interface. The authors have identified similarities between DEVS and SysML [18]. Combining alternative SysML diagrams to define all aspects of coupled and atomic DEVS models facilitated the definition of a SysML profile. Moreover, employment of proper stereotypes and constraints have made enrichment of SysML models feasible.

There are several simulation environments for DEVS, as well as attempts to provide generic representations of DEVS models, mainly targeting DEVS simulators interoperability. Many of them, as the one presented in [19], are based on XML representation of DEVS entities. These efforts provided the foundations for the definition of a reference DEVS meta-model in XMI [7].

A SysML profile and corresponding constraints for DEVS simulation framework have been defined and implemented in MagicDraw [17] modelling tool, enabling proper enrichment of SysML system models (step 1). This required to locate the exact parts of the SysML models, where DEVS-specific attributes should be added. Also, specification of behavioral details in a generic, yet precise manner has been a difficult task, in contrary to structural aspects. Fig. 2 illustrates a screen shot of a *state machine diagram* defining an aspect of the DEVS simulation model for a *regiment* component in a *battle* simulation example.

A MOF 2.0 meta-model for DEVS models has been defined, based on previous attempts to standardize DEVS model representations in XML format and mainly [20]. The meta-model is a central element in the overall approach and needs to be finalized first. Changes on the meta-model would affect both the transformation from system models to DEVS models and from DEVS models to executable format. Additionally, a QVT transformation that generates DEVS models from enriched SysML models has been defined (step 2). The definition of the whole set of QVT relations, constituting the transformation, required an effort investment that is not returned for few, simple models. However, applying the transformation in larger, complex models justifies such an approach. Enriched system models have been extracted from MagicDraw in XMI format and transformed to the respective DEVS models. Fig. 3 presents a part of the DEVS model that is generated from the SysML model shown in Fig. 2.

```
<DEVS_ATOMIC>
 <MODEL_NAME text="regiment"/>
 <INPUTS>...</INPUTS>
 <OUTPUTS>...</OUTPUTS>
 <STATES>
  <STATE_SET>
   <STATE_SET_NAME text="STATE SET"/>
   <STATE_SET_VALUES>
    <STATE_SET_VALUE text="idle" initial="true"/>
    <STATE_SET_VALUE text="attackingWithInfantry"/>
    <STATE_SET_VALUE text="attacking"/>
    <STATE_SET_VALUE text="attakingWithArtillery"/>
    <STATE_SET_VALUE text="waiting"/>
    <STATE_SET_VALUE text="waitingArtillery"/>
   </STATE_SET_VALUES>
  </STATE_SET>
  <STATE_VARIABLES>...</STATE_VARIABLES>
 </STATES>
 <INTERNAL_TRANSITION_FUNCTION>
  <CONDITIONAL_FUNCTION>
   <STATE_CONDITION text="attacking"/>
   <TRANSITION_FUNCTION>
    <NEW_STATE text="waiting"/>
    <STATE_VARIABLE_UPDATES/>
   </TRANSITION_FUNCTION>
  </CONDITIONAL_FUNCTION>
  ...
 </INTERNAL_TRANSITION_FUNCTION>
 <OUTPUT_FUNCTION>
  <CONDITIONAL_OUTPUT_FUNCTION state="attacking">
   <PORT_OUTPUTS>
    <SEND port="InfantryAttack">
     <VALUE type="Integer" value="1"/>
    </SEND>
    <SEND port="ArtilleryAttack">
     <VALUE type="Integer" value="1"/>
    </SEND>
   </PORT_OUTPUTS>
  </CONDITIONAL_OUTPUT_FUNCTION>
  ...
 </OUTPUT_FUNCTION>
 <TIME_ADVANCE_FUNCTION>
  <CONDITIONAL_TIME_ADVANCE>
   <STATE_CONDITION text="attacking"/>
   <TIME_ADVANCE>
    <VALUE type="Real" value="0.1"/>
   </TIME_ADVANCE>
  </CONDITIONAL_TIME_ADVANCE>
  ...
 </TIME_ADVANCE_FUNCTION>
 <EXTERNAL_TRANSITION_FUNCTION>
  ...
 </EXTERNAL_TRANSITION_FUNCTION>
</DEVS_ATOMIC>
```

Fig. 3. DEVS model, derived from the enriched SysML model

In order to implement the third step of the methodology, the approach presented in [21] has been utilized. This approach was selected, since it supports an XML-based language, named XLSC, for describing DEVS models that can be executed into a DEVSJava simulator, promoting model transformation. An XSL transformation from DEVS models in XMI format to executable XLSC simulation models in XML format has also been defined and tested. Fig. 4 presents a part of the *regiment* model in XLSC (executable) format.

```
<atomicModel name="regiment">
 <statePart>
  ...
 </statePart>
 <propertiesPart/>
 <portsPart>
  ...
 </portsPart>
 <functionsPart>
  ...
  <internalTransitionFunction>
   <action>
    <if>
     <condition>
      <equal>
       <retrieve state="phase"/>
       <string>attacking</string>
      </equal>
     </condition>
     <then>
      <update state="phase">
       <string>waiting</string>
      </update>
     </then>
    </if>
    ...
  </internalTransitionFunction>
 </functionsPart>
</atomicModel>
```

Fig. 4. DEVS executable model in XLSC format

VI. Conclusions

Following the proposed guidelines and the three-step methodology, tools of simulating SysML models with DEVS have been successfully applied and implemented. The DEVS-SysML profile and DEVS MOF meta-model were proven to the fundamental elements of the approach, enabling in practice automated code generation. The choice of DEVS as a simulation methodology also derived from the proposed guidelines. Standard model transformation languages, such as QVT, also proved to be efficient, provided that corresponding XML-based tools are available for simulation. Furthermore, they promote interoperability between different simulation methodologies, provided corresponding MOF meta-models are defined.

Future work include the application of tools developed in different domains, such as military and information system simulation, and the support of simulation model libraries. The integration of the proposed DEVS-SysML profile with system design profiles is also under investigation. The definition MOF metamodels for different simulation environments and the exploitation of a common SysML profile, serving all of them, will also be addressed.

VII. Acknowledgements

The authors would like to thank Nicolas Meseth, Patrick Kirchhof, and Thomas Witte for their valuable help. Not only they provided us with their XLSC prototype interpreter, but also they eagerly answered every question we posed.

References

[1] OMG, "Systems Modeling Language (SysML) Specification. Version 1.0," September 2007.

[2] ——, "Model Driven Architecture. Version 1.0.1," *Available online via http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf*, June 2003.

[3] L. McGinnis and V. Ustun, "A simple example of SysML-driven simulation," in *Winter Simulation Conference (WSC), Proceedings of the 2009*. IEEE, 2009, pp. 1703–1710.

[4] R. Wang and C. Dagli, "An executable system architecture approach to discrete events system modeling using SysML in conjunction with colored petri nets," in *IEEE Systems Conference 2008*. Montreal: IEEE Computer Press, April 2008, pp. 1–8.

[5] W. Schamai, "Modelica Modeling Language (ModelicaML): A UML Profile for Modelica," Tech. Rep., 2009. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-20553

[6] OMG, *SysML and Modelica Integration*, Dec. 2008. [Online]. Available: http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-modelica:sysml_and_modelica_integration

[7] G. D. Kapos, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos, "An integrated framework for automated simulation of SysML models using DEVS," 2012, under Submission.

[8] B. P. Zeigler, H. Praehofer, and T. Kim, *Theory of Modeling and Simulation*, 2nd ed. Academic Press, 2000.

[9] E. Huang, R. Ramamurthy, and L. F. McGinnis, "System and simulation modeling using SysML," in *WSC '07: Proceedings of the 39th conference on Winter simulation*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 796–803.

[10] O. Schonherr and O. Rose, "First steps towards a general SysML model for discrete processes in production systems," in *Proceedings of the 2009 Winter Simulation Conference*, Austin, TE, USA, December 2009, pp. 1711–1718.

[11] R. Peak, R. Burkhart, S. Friedenthal, M. Wilson, M. Bajaj, and I. Kim, "Simulation-based design using SysML part 1: A parametrics primer," in *INCOSE Intl. Symposium*, San Diego, CA, USA, 2007, pp. 1–20.

[12] R. Peak, C. J. Paredis, and D. R. Tamburini, "The composable object (COB) knowledge representation: Enabling advanced collaborative engineering environments (CEEs), COB requirements & objectives (v1.0)," Georgia Institute of Technology, Atlanta, GA, Technical Report, Oct. 2005.

[13] D. R. Tamburini, "Defining executable design & simulation models using SysML," *Available online via http://www.pslm.gatech.edu/topics/sysml/*, March 2006.

[14] A. A. Kerzhner, J. M. Jobe, and C. J. J. Paredis, "A formal framework for capturing knowledge to transform structural models into analysis models," *Journal of Simulation*, vol. 5, no. 3, pp. 202–216, 2011.

[15] Modelica, *The Modelica Language Specification Version 3.2*, revision 1 ed., February 2012.

[16] IBM, "Rational Software Modeler," *Available online via http://www.ibm.com/developerworks/rational*, 2010.

[17] MG, *SysML Plugin for Magic Draw*, 2007.

[18] M. Nikolaidou, V. Dalakas, L. Mitsi, G.-D. Kapos, and D. Anagnostopoulos, "A SysML profile for classical DEVS simulators," in *Proceedings of the Third International Conference on Software Engineering Advances (ICSEA 2008)*. Malta: IEEE Computer Society, October 2008, pp. 445–450.

[19] S. Mittal, J. L. Risco-Martín, and B. P. Zeigler, "DEVSML: Automating DEVS execution over SOA towards transparent simulators," in *DEVS Symposium, Spring Simulation Multiconference*. ACIMS Publications, March 2007, pp. 287–295.

[20] J. Martín, S. Mittal, M. López-Peña, and J. De la Cruz, "A W3C XML schema for DEVS scenarios," in *Proceedings of the 2007 spring simulation multiconference-Volume 2*. Society for Computer Simulation International, 2007, pp. 279–286.

[21] N. Meseth, P. Kirchhof, and T. Witte, "XML-based DEVS modeling and interpretation," in *SpringSim '09: Proceedings of the 2009 Spring Simulation Multiconference*. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 1–9.