Simulating SysML Models: Overview and Challenges

Mara Nikolaidou, George-Dimitrios Kapos, Anargyros Tsadimas,
Vassilios Dalakas and Dimosthenis Anagnostopoulos
Department of Informatics and Telematics
Harokopio University of Athens
Athens, GREECE
{mara, gdkapos, tsadimas, vdalakas, dimosthe}@hua.gr

Abstract – *SysML language, proposed by OMG, is a commonly accepted standard to model and study systems-of-systems (SoSs). It provides the means to depict SoS components and their behavior in a multi-layer fashion and explore alterative architectures for their design. To validate SysML models in terms of performance criteria, simulation is usually the preferred method employed. To this end, different SysML diagrams are utilized, while numerous simulation methodologies and tools are employed. There are many efforts targeting simulation code generation from SysML models. Model-based system engineering concepts are adopted in most of these to generate simulation models from SysML models. Nevertheless, this process is not standardized, although most of current approaches tend to follow the same steps, even if they employ different tools. The scope of this paper is to provide a comprehensive understanding of the similarities and differences of existing approaches and identify current challenges in fully automating SysML model simulation process.*

Keywords: SysML, simulation, automated code generation, model transformation, model-based system engineering.

1 Introduction

SysML ([1]) is a language commonly used for model-based system design (MBSD). It is an Object Management Group (OMG) standard that supports specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. It provides discrete diagrams to describe system structure and components, to explore allocation policies crucial for system design and to identify design requirements. It is widely applied for systems-of-systems (SoS) engineering [2].

SysML is a general-purpose language, facilitating modeling of any system or system-of-systems. UML profiling mechanism can be employed to restrict or extend SysML features to serve a specific domain, as for example real-time and embedded systems [3] or information systems [4]. These profiles, accompanied with specific plugins, can be executed within UML modeling tools (such as Magic Draw [5] or IBM Rational Modeler [6]) are capable of producing valid system models for the specific domain, according to the profile and UML/SysML specifications.

As simulation is a common method for estimating the performance of systems, there is currently strong interest in generating simulation code from SysML models. Recent efforts (as for example [7], [8], [9]) provide the ability to generate executable simulation code of different simulation languages or environments (as for example Arena, Modelica or DEVS). In most of these efforts, profiling mechanism is used to embed simulation properties in SysML models. Simulation-specific profiles are employed in a popular modeling tool, as Magic Draw or IBM Rational Modeler mentioned above, to annotate SysML models with simulation properties appropriate for the specific simulator. Afterwards, the corresponding enriched SysML models are transformed to executable simulation code for the specific environment. Model transformation languages, such as ATL [10] and QVT [11], are often utilized to transform SysML models to simulation models represented in XMI, an XML representation language for UML/SysML models. In such cases, model-based engineering techniques are adopted and model-driven software generation is employed according to MDA standards [12]. The benefit of employing such standards to accomplish simulation code generation is the fact that all the steps leading to the creation of executable simulation models are independent of the tools used for SoS modeling and engineering and the specific system domain and the methodology employed and software created are only restricted by the simulation methodology/environment.

Although the process of generating simulation code is similar in the aforementioned approaches, it is not standardized, while there are still restrictions in fully automating the simulation code generation process related to the system domain, the existence of pre-defined simulation libraries of the specific domain and the characteristics of the simulation methodology/environment. To this end, a thorough overview of different approaches for simulating SysML models is conducted in this paper, in order to identify similarities and basic differences, in a effort to standardize the process of simulating SysML models taking into account that different tools and methods

are to by used. The approaches studied were selected based on two criteria:

a) They are based on model-based system engineering concepts and apply MDA standards as discussed above to ensure compatibility with current SoS engineering standards. Manual or semi-automated generation of executable simulation models from SysML models can be cumbersome, tedious and error-prone. In addition, building custom tools for this purpose restricts reusability and interoperability with other simulation platforms.

b) Different simulation methodologies/environments are utilized to ensure the generality of the deduced conclusions. The simulation techniques selected may serve different system domains, implement either discrete or continuous simulation and utilize different SysML diagrams to integrate simulation characteristics into SysML models.

Besides the simulation code generation process, the system validation process after the simulation is completed is also explored. In this case, simulation data are used to verify performance requirement defined during system design. There are approaches, such as [13], [14], [15], providing comprehensive solutions in assisting the system engineer identifying system design conflicts or drawbacks. However, the automated and transparent integration of the system validation process within the SysML model used of system design with the modeling tool remains a challenge.

The rest of the paper is organized as follows: In Section 2 related work is discussed. In Section 3 existing approaches for simulating SysML models are discussed, while their basic characteristics are explored in a comparative study. Challenges in automated simulation code generation and system validation are identified in Section 4.

2 Related work

There are many efforts that employ SysML for model-based system design in different domains. Recently, SLIM [16], a commercial collaborative model-based systems engineering workspace that uses SysML as the front-end for orchestrating system engineering activities from the early stages of system development, is available from Intercax. The SysML-based system model serves as a unified, conceptual abstraction of the system, independent of the specific design and analysis tools that shall be used in the development process. It is designed to provide plugins to integrate the system model to a variety of design and analysis tools. Until now, only the integration of SysML and other model repositories, such as product lifecycle management (PLM) tools is implemented. Integration with MATLAB/Simulink, Mathematica and OpenModelica is offered in a variety of commercial tools, but automated simulation code generation is not implemented yet.

Focusing on the description of specific domain entities, specialized SysML profiles are introduced, while the need to manage performance requirements during system design is also addressed. SysML provides the means for requirement description, while there are efforts, such as [4], [15], [17] focusing on requirement verification described using SysML.

In any case, to validate SysML models in terms of performance, they should be simulated first. Apparently SysML supports a variety of diagrams describing system structure and states, necessary to perform simulation, which are utilized by different approaches ([18], [19]). In most cases, SysML models defined within a modeling tool are exported in XMI format and, consequently, transformed into simulator specific models to be forwarded to the simulation environment. To embed simulation-specific properties within SysML models, profiles are introduced. In all cases, stereotypes and constraints defined within the profile are related to the simulation platform employed ([7], [8], [9], [15], [18], [19]). Simulation model validity may be ensured by applying constraints in the models produced by the profile using declarative languages, as OCL ([7], [8], [3], [20]) or even java plugins ([14]).

Some of them are general ([9], [18]) constraint only by the simulation platform and facilitating simulation of any kind of systems. In such case both the system structure and the system behavior described in terms of the simulation environment must by defined. Both discrete event (for example [9]) and continuous (for example [13]) simulation environments are supported, utilizing different SysML diagrams, e.g activity and state diagrams in the first case and parametric diagrams in the second case to model system behavior. In the case of general approaches, simulation code generation is usually not fully automated in terms of system behavior, which is restricted in the profile in term of functionality and expressiveness.

Most approaches are focused on a specific system domain, while profiles contain stereotypes to describe specific domain components, while their behavior is prescribed in simulation libraries contained in the simulation environments selected. Popular examples of such systems are presented for example in [17] for embedded systems simulated using Modelica or in [7] for manufacturing assembly systems simulated using Arena. Depending on the nature and specific characteristics of the specific domain, there is a diversity of ways proposed to simulate SysML models, utilizing different diagrams.

Existing approaches may also be grouped in an alternative fashion, depending on whether or not they are utilizing current model-driven software engineering standards for simulation code generation. Custom tools, not utilizing existing standards, although flexible and fast, do not promote model transformation validation, while they restrict reusability and interoperability with other

simulation platforms. Many approaches are grouped in this category, such as those presented in [3], [4], [20].

In order to follow model-driven code generation principles and facilitate SysML to simulation model transformation, a meta-model describing the simulation entities supported by the simulation must be defined. The existence of such a simulation meta-model is imperative to facilitate the transformation of SysML models described in XMI format into simulation models [21]. To ensure compatibility with UML/SysML related standards, MOF 2.0, the meta-model defined by OMG to define them must be used also for simulation meta-model description. MOF 2.0 compliance enables the application of standard languages, such as ATL or QVT to program SysML-to-simulation model transformation. Providing a MOF 2.0 metamodel in XMI format for a simulation methodology or tool, such as those defined for Modelica [22] or DEVS [23] enhances the transformation process and facilitates the usage of the specific simulation methods for SysML model simulation. The transformation needs to be defined only once for a pair of domain and simulation environment.

3 A Comparative Overview of SysML Model Simulation Approaches

Out of the extensive literature for SysML model simulation, the approaches presented in the following were chosen taking into account the following criteria: a) they are based on model-driven software engineering, b) they support all the steps discussed, i.e. profile definition, automated simulation code generation and system validation and c) they utilize different simulation environments.

MARTE Profile and Related Tools

MARTE is a UML profile proposed by the OMG [24] in 2009, supporting model-based design of real-time and embedded systems. It focuses on the performance and scheduling properties of real-time systems. Performance and scheduling requirements are modeled as constraints defined using VSL, a language for formulating semantically well-formed algebraic and time expressions.

After SysML standardization, there are numerous efforts to combine SysML and MARTE profiles (for example [3], [25], [26], [27]). Basically they focus on integrating SysML requirements and VSL language, used for their specification. [3] focuses on avionics and surveillance embedded systems, while [26] on embedded software systems. VSL well-defined semantics enable the automated verification of defined requirements/constraints with external tools.

In [26] and [27] the effort presented is focuses on generating executable code in SystemC, a language for describing executable software for embedded systems based using model transformation techniques. Furthermore, the same methodology is suggested to provide executable models for Promela/SPIN model checking environment, MATLAB Stateflow simulation environment. No detailed information on MOF 2.0 based meta-models for all these environments are provided, though MDA principles are adopted by the authors. In [27], the MDEReqTraceTool, currently under development, is proposed to integrate requirement verification information within SysML system models by updating corresponding SysML requirement verification matrixes. Such a feature will enable the MARTE requirements verification using external tools, in a transparent fashion for the system designer working with MARTE/SysML models.

CASSI Tool

In [4] SysML extensions were proposed for information system design, which are implemented within the context of a custom, in-house tool called CASSI. CASSI targets information system integration, while three different design views are supported, depicted using SysML external and internal block diagrams. The behavior of system components is described within CASSI using sequence diagrams, transformed to a simulation model based on Petri-nets, which is executed by an external simulator. Although based on MDA principles, existing standards and tools are not utilized. CASSI is based entirely on custom tools.

As described on [28], information system configurations defined using CASSI are evaluated using simulation to verify performance and availability requirements. NFR verification is performed by the system designer using external tools by the aid of Service Level Objective (SOL) concept, while system validation and requirement verification results are not integrated within SysML system model.

TTool Toolkit and Related Efforts

TTool Toolkit (http://ttool.telecom-paristech.fr) integrates numerous tools targeting real-time embedded system engineering. AVATAR SysML profile targeting safety and security properties of embedded systems [31]. TEPE, a graphical expression language based on SysML parametric diagrams, is introduced for representing requirements making them amenable to automated verification [29]. The profile also enables the definition of system behavior through state machine diagrams. Model verification is performed using a constraint language called UPPAAL (based on OCL), to ensure system model validity before simulating them [30]. DIPLODOCUS, a simulation engine targeting on System-on-Chip design, is integrated in TTool. It is based on Y-Chart approach simulation approach timed-automata. TTool also supports The IFx toolkit3 [32] also provides simulation capabilities within TTool framework. In this case also, although model-driven engineering concepts are introduced to automatically generate simulation code based on predefined libraries for

the domain of embedded systems, no MOF 2.0 compatible meta-model is defined for the simulation or model checking environments. Requirement verification is facilitated by external tools, such as IFx toolkit3.

SysML-to-Arena Transformation Tools

In [7], manufacturing line system models defined in SysML and transformed using ATL to be simulated using Arena simulation software. With the definition of a SysML profile Arena-specific properties to represent manufacturing systems are incorporated within SysML block definition and activity diagrams [33]. Corresponding ARENA simulation libraries are incorporated with ARENA environment, and properly instantiated to construct the simulation model executed within ARENA tool. As far as simulation is concerned only system structure is defined in SysML diagrams. System simulation behavior is defined within ARENA manufacturing system libraries. SysML-to-ARENA model transformation is performed using ATL based on model-based software engineering principles, while a corresponding MOF-based meta-model for ARENA manufacturing system libraries is defined. The exploitation of simulation output towards system model validation is not discussed.

SysML4Modelica Project

The SysML4Modelica profile endorsed by OMG [8] enables the transformation of SysML models to executable Modelica simulation code. To embed simulation capabilities within SysML, ModelicaML profile is used [22]. QVT is used for the transformation of SysML models defined using ModelicaML profile to executable Modelica models. A corresponding MOF 2.0 meta-model for Modelica is defined. The overall approach is fully compatible with model-driven engineering concepts, making it suitable of efficient SoS engineering. In [17] focus is given on using the SysML4Modelica profile foe embedded systems engineering.

In the proposed profile, SysML requirement entity is extended with testable characteristics. Testable requirements are associated to conditions under which the requirement is verified with the use of experiments or test cases. Verification conditions are defined as part of a test case, which in turn may be simulated using Modelica simulation language in external simulators to ensure that a design alternative satisfies related requirements [13]. Requirement verification is performed in an external modelica tool (MathModelica) through visual diagrams created during simulation.

The only possible limitation of this approach is that it is tailor-made to the Modelica in order to describe the requirements and the verification process together.

DevSys Framework

The authors have proposed an integrated framework for the simulation of SysML model using DEVS [9]. DEVS SysML profile for the enrichment of SysML models with all required properties to generate the classical DEVS simulation models [34]. Block definition and internal block diagrams are utilized to describe system structure, while state machine, activity and parametric diagrams are utilized to define system behavior for simulation purposes. To this end, MOF 2.0 meta-model for DEVS is proposed and applied for the definition of a standards-based QVT transformation of enriched SysML models to DEVS models that are consequently transformed to executable DEVS code [35]. Additionally, performance-related attributes are defined of all system components, calculated during simulation and embedded within the SysML model after the completion of the simulation [14]. In such case, the performance-related attributes are restricted to those defined in the profile, to ensure the automated generation of the corresponding simulation code. The profile is not restricted to a specific system domain, enabling the simulation of any system described in SysML following DEVS behavioral model. Thus, the main restriction of this approach is that the system designer should be aware of DEVS methodology and concepts to properly define system behavior. Constraints defined in OCL and Java plugins are available in the profile to ensure model validity before simulation.

The combination of DEVSys framework with EIS profile for information system design in order to simulate information system models defined using the profile is presented in [14]. In this case, information system component libraries were implemented within DEVS, while SysML-to-DEVS QVT transformation was utilized only to generate the simulation code corresponding to system structure. System behavior was already implemented in DEVS libraries.

Summary

The basic features of the reviewed approaches are summarized in Table 1. A variety of simulators are utilized, selected usually based on the system domain and their popularity. Custom solutions tend to be avoided. None of the presented works fully covers all features effectively. However, the adaptation of model-based engineering principles and the utilization of standard language to transform SysML-to-Simulation model is clearly gaining momentum. Most of the approach, even if defined to be general, focus on a specific system domain and provide corresponding model libraries within the simulation environment to simply the transformation process. In most case model simulation is closely related to corresponding requirement verification, while most of the approaches tackle with requirement verification as well, if though in many case this process in performed outside the SysML modeling tool. Finally, the automated simulation code generation capabilities offered are constantly increasing.

Table 1: A Comparative Overview of SysML Simulation Approaches

Profile	Features Offered						
	System domain	*Simulator*	*Profile Characteristics*	*Transformation language*	*MDA conformance*	*Code generation support*	*SysML model validation/ requirement verification*
MARTE Profile	Real-Time Embedded Systems	Non-Specified	- Focus on performance and time requirement description and verification utilizing VSL	Non Specified	Medium	Non specified	MDEReqTrace Tool, currently under development, integrates requirement verification data from external tools within SysML system models
CASSI Tool	Information Systems	Petri-Nets	- Focus on describing performance requirements - System behavior is described using Sequence diagrams	Non Specified	Low	Semi automated	Requirement verification performed by an custom external tool
TTool Toolkit	Real-Time Embedded Systems	Y-Chart, Timed-Automata	- Focus on requirement description using TEPE - System behavior is described using State Machine diagrams	Non Specified	Medium	Fully automated	Requirement verification performed by external tools
SysML to Arena Tools	Manufacturing Line Systems	Arena	- Focus on the description of the specific domain to incorporate simulation-related characteristics	ATL	High	Fully automated	Not Specified
SysML4 Modelica	General (emphasis on real-time systems)	Modelica	- Focus on describing performance requirements - System behavior under exploitation is defined as Test Cases using Modelica ML	QVT	High	Fully automated	Requirement verification performed by external tools
DEVSys Framework	General (case study: Information Systems)	DEVS	- Focus on embedding simulation output within SysML models - System behavior is described using State Machine, parametric and Activity diagrams	QVT	High	Fully automated	Requirement verification performed within SysML models (in cooperation with EIS profile)

4 Challenges Identified

Having in mind existing approaches, it is evident that there is a strong interest in simulating SysML models in an automated fashion to serve SoS engineering and especially SoS design. Since different system domains should be effectively supported, it is expected that different simulation methods and tools will be employed. Though, it is imperative that a standardized methodology/framework, based on OMG standards, should be proposed to guide experts to develop tools targeting specific domains and simulation environments. Most recent approaches seam to follow the same basic steps:

a) Define simulation/domain specific profiles. In this process, efforts should concentrate on define simulator-specific profiles that may be combined with domain specific profiles. Furthermore, the exploration of a simulator-agnostic profile is suggested for discrete-event and continuous simulators respectively, taking into account that existing approaches utilize the same SysML diagrams.

b) Transform SysML to simulation models in a standardized fashion, utilizing languages as QVT and ATL. Simulator-specific profiles should be accompanied by corresponding MOF-based meta-models for the corresponding simulators. The definition of such meta-models openly available may also promote simulator interoperability. Corresponding initiatives, as those employed by Modelica and DEVS community are already successful.

c) Utilize simulation output to validate SysML models and verify corresponding requirements defined in such models. In order to simplify requirement verification process, we endorse the suggestion of SLIM to conduct requirement verification within SysML modeling tools, independently of the simulation methods and tools. The incorporation of simulation results within the SysML model should be facilitated for this purpose. Such enhancements simplify the evaluation process, allowing the system designer to focus on the examination of the unverified requirements and, consequently, the detection of the necessary solution re-adjustments.

As derived from the examination of existing approaches, there are two key issues in requirements verification during model-based system design that have not been fully addressed: (a) the estimation of system models behavior in a generic and -at the same time- automated manner, and (b) the designation of the requirements that have not been verified in the original system model. Precisely, although simulation has been identified as an appropriate technique for the estimation of system models' performance, obtained simulation results should be incorporated within the original system model and comparison against the predefined, performance-related, requirements should be performed within the modeling environment.

References

[1] OMG, Systems Modeling Language (SYSML) Specification, Version 1.3 (June 2012). URL http://www.omg.org/spec/SysML/1.3/PDF

[2] INCOSE, Systems Engineering Handbook, version 3.2.2 Edition, International Council on Systems Engineering, San Diego, CA, USA, 2012.

[3] Quadri, I.R., Sadovykh, A, Indrusiak, L.S, "MADES: A Mixed SysML/MARTE methodology for real-time and embedded avionics systems," ERTS 2012 Conference in Toulouse, France, Feb. 2012.

[4] S. Izukura, K. Yanoo, T. Osaki, H. Sakaki, D. Kimura, J. Xiang, Applying a model-based approach to IT systems development using SysML extension, in: MoDELS, Vol. 6981 of Lecture Notes in Computer Science, Springer, 2011, pp. 563–577. URL http://dx.doi.org/10.1007/978-3-642-24485-8

[5] NoMagic, SysML Plugin for Magic Draw. URL http://www.nomagic.com/products/magicdraw-addons/ sysml-plugin.html

[6] IBM, Rational Software Modeler. URL http://www.ibm.com/developerworks/rational

[7] O. Batarseh, L. F. McGinnis, System modeling in SysML and system analysis in ARENA, in: Proceedings of the Winter Simulation Conference, WSC '12, 2012, pp. 258:1–258:12.

[8] OMG, SysML-Modelica Transformation (SyM) (Nov. 2012).URL http://www.omg.org/spec/SyM/1.0/PDF/

[9] G.-D. Kapos, V. Dalakas, M. Nikolaidou, D. Anagnostopoulos, An integrated framework for automated simulation of SysML models using DEVS, Simulation 90 (6) (2014) 717–744.

[10] Atlas Transformation Language. URL https://eclipse.org/atl/

[11] OMG, MOF 2.0 Query/View/Transformation Language ver. 1.1, Jan. 2011. URL http://www.omg.org/spec/QVT/1.1

[12] OMG, Model-Driven Architecture. URL http://www.omg.org/mda/

[13] W. Schamai, P. Helle, P. Fritzson, C. J. J. Paredis, Virtual verification of system designs against system requirements, in: Proceedings of the 2010 international conference on Models in software engineering, MODELS'10, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 75–89.

[14] A Tsadimas, GD Kapos, V Dalakas, M Nikolaidou, D Anagnostopoulos, "Integrating Simulation Capabilities into SysML for Enterprise Information System Design", IEEE 8th Conference on SoSE, Adelaide, Australia, June 2014.

[15] J.-F. Petin, D. Evrot, G. Morel, P. Lamy, Combining SysML and formal methods for safety requirements verification, in: 22nd International Conference on Software & Systems Engineering and their Applications, Paris, France, 2010.

[16] M. Bajaj, D. Zwemer, R. Peak, A. Phung, A. Scott, M. Wilson, Slim: collaborative model-based systems engineering workspace for next-generation complex systems, in: Aerospace Conference, 2011 IEEE, 2011, pp. 1–15.

[17] A. A. Kerzhner, J. M. Jobe, C. J. J. Paredis, A formal framework for capturing knowledge to transform structural models into analysis models, Journal of Simulation 5 (3) (2011) 202–216.

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

[19] O. Schonherr, 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, 2009, pp. 1711–1718.

[20] S. Chouali, A. Hammad, H. Mountassir, Assembling components using sysml with non-functional requirements, Electronic Notes in Theoretical Computer Science 295 (0) (2013) 31 – 47, Proceedings the 9th International Workshop FESCA.

[21] M. Nikolaidou, G.-D. Kapos, V. Dalakas, D. Anagnostopoulos, Basic Guidelines for Simulating SysML Models: An Experience Report, in: Proc. Seventh Int. Conf. on System of Systems Engineering (SoSE) 2012, 2012, pp. 95–100.

[22] W. Schamai, Modelica Modeling Language (ModelicaML): A UML Profile for Modelica, Tech. rep. (2009).

[23] S. Mittal and J. L. Risco Martín, Netcentric System of Systems Engineering with DEVS Unified Process, CRC Press, 2013.

[24] OMG, UML profile for MARTE: Modeling and analysis of real-time embedded systems specification, version 1.0, Nov 2009.

[25] H. Espinoza, D. Cancila, B. Selic, S. Gerard, Challenges in combining SysML and MARTE for model-based design of embedded systems, in: ECMDA-FA, Vol. 5562 of Lecture Notes in Computer Science, Springer, 2009, pp. 98–113.

[26] Rota Sena Marques, M.; Siegert, E.; Brisolara, L., "Integrating UML, MARTE and sysml to improve requirements specification and traceability in the embedded domain," Industrial Informatics (INDIN), 2014 12th IEEE International Conference on, pp.176,181, 27-30 July 2014.

[27] Marcello Mura, Amrit Panda, Mauro Prevostini, "Executable Models and Verification from MARTE and SysML: a Comparative Study of Code Generation Capabilities". MARTE Workshop (2008)

[28] D. Kimura, T. Osaki, K. Yanoo, S. Izukura, H. Sakaki, A. Kobayashi, "Evaluation of IT systems considering characteristics as system of systems", 6th International Conference o System of Systems Engineering (SoSE), 2011.

[29] D. Knorreck, L. Apvrille, P. de Saqui-Sannes, TEPE: A sysml language for time-constrained property modeling and formal verification, SIGSOFT Softw. Eng. Notes 36 (2011) 1–8.

[30] G. Behrmann, A. David, K. Larsen, J. Hakansson, P. Petterson, W. Yi, M. Hendriks, UPPAAL 4.0, in: Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on, 2006, pp. 125–126.

[31] G. Pedroza, L. Apvrille, D. Knorreck, Avatar: A SysML environment for the formal verification of safety and security properties, in: NOTERE 2011, 11th Annual International Conference on, 2011, pp. 1–10.

[32] I. Ober, S. Graf, I. Ober, Validating timed UML models by simulation and verification, International Journal on Software Tools for Technology Transfer 8 (2) (2006) 128–145.

[33] L. F. McGinnis, E. Huang, K.S. Kwon, V. Ustun, "Ontologies and simulation: a practical approach", Journal of Simulation, 08/2011; 5:190-201.

[34] B. P. Zeigler, H. S. Sarjoughian, Introduction to DEVS Modeling and Simulation with JAVA (2003).

[35] G.-D. Kapos, V. Dalakas, A. Tsadimas, M. Nikolaidou, D. Anagnostopoulos, Model-based system engineering using SysML: Deriving executable simulation models with QVT, in: Systems Conference (SysCon), 2014 8th Annual IEEE, IEEE, 2014, pp. 531–538.