ACSIJ Advances in Computer Science: an International Journal, Vol. 5, Issue 4, No.22 , July 2016
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Challenges in SysML Model Simulation

Mara Nikolaidou, George-Dimitrios Kapos, Anargyros Tsadimas, Vassilis Dalakas and Dimosthenis Anagnostopoulos

Department of Informatics and Telematics
Harokopio University of Athens
70, El. Venizelou Str, Athens, GREECE
{mara, gdkapos, tsadimas, vdalakas, dimosthe}@hua.gr

Abstract

Systems Modeling Language (SysML) is a standard proposed by the OMG for systems-of-systems (SoS) modeling and engineering. To this end, it provides the means to depict SoS components and their behavior in a hierarchical, multi-layer fashion, facilitating alternative engineering activities, such as system design. To explore the performance of SysML, simulation is one of the preferred methods. There are many efforts targeting simulation code generation from SysML models. Numerous simulation methodologies and tools are employed, while different SysML diagrams are utilized. 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 models 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), as it facilitates modeling of any system or system-of-systems. 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].

On the one hand, UML profiles 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]) and are capable of producing valid system models for the specific domain, based on the profile specifications.

On the other hand, 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 for different simulation languages or environments (as for example Arena, Modelica or DEVS, respectively). In most of these efforts, the UML profiling mechanism is used to embed simulation properties into SysML models. Simulation-specific profiles are employed in a popular modeling tool, such as Magic Draw or IBM Rational Modeler mentioned above, to annotate SysML models with simulation properties appropriate for the specific simulator. Afterwards, 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 system engineering techniques are adopted and model-driven software generation is employed according to MDA principles ([12], [13]). The main benefit of employing such standards to accomplish simulation code generation is that all steps leading to the creation of executable simulation models are independent of the tools used for SoS modeling and engineering. Hence, the specific system domain, the methodology employed and the created software are only restricted by the simulation methodology/environment.

Although, in the aforementioned approaches the process of generating simulation code is similar, it is not standardized. Furthermore, there are still restrictions in fully automating the simulation code generation process related to the system domain, the existence of pre-defined simulation libraries and the characteristics of the simulation methodology/environment. To this end, a thorough overview of different approaches for simulating SysML models is presented in [14]. The approaches studied were selected based on two criteria:

a) To be based on model-based system engineering concepts and to apply MDA standards as discussed

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

above to ensure compatibility with current SoS engineering standards.

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

In the following, we emphasize on the challenges that should be explored, based on the experience accumulated from existing approaches so far, in an effort to standardize the process of simulating SysML models, taking into account that different tools and methods should be used.

Besides the generation process related with the simulation code, the incorporation of simulation output into the system after the completion of the simulation, is also explored. In this case, simulation data are used to validate system models. There are approaches, such as [13], [15], providing comprehensive solutions in assisting the system engineer to identify system design conflicts or drawbacks. However, the automated and transparent integration of the system validation process within the SysML model used for the system design with the modeling tool remains a challenge.

The rest of the paper is organized as follows: In Section 2 an overview of existing approaches for simulating SysML models based on MDA principles is presented, while their prominent features are summarized in a comparative study. Challenges and future directions in automated simulation code generation and system validation are identified in Section 3.

2. SysML Model Simulation Overview

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. Integration with MATLAB/Simulink, Mathematica and OpenModelica is planned for a variety of commercial tools, but automated simulation code generation is not implemented yet. SysML also 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, which are utilized by different simulation 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 ([21]).

There are general approaches ([9], [13], [18]) constrained only by the simulation platform and facilitating simulation of any kind of systems. In this case, simulation code generation is usually not fully automated in terms of system behavior, which is restricted in the profile in terms of functionality and expressiveness. Though, most of existing approaches are focused on a specific system domain. In this case, corresponding 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.

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 ([3], [4], [20]).

In order to follow model-driven code generation principles, the existence of a simulation meta-model is imperative for the transformation of SysML models described in XMI format into simulation models [9]. To ensure compatibility with UML/SysML related standards, MOF 2.0, the meta meta-model proposed by the OMG to

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

define them, should be used for simulation meta-model description. MOF 2.0 compliance enables the application of standard languages, such as ATL or QVT to define SysML-to-Simulation model transformation. Providing a MOF 2.0 meta-model for a simulation methodology or tool, such as those defined for Modelica [22] or DEVS [23], enhances the transformation process of SysML-to-simulation models. The transformation needs to be defined only once for a pair of domain and simulation environment.

A thorough overview of existing SysML model simulation approaches is provided in [14]. In the following, we focus on specific approaches having in mind the following criteria: a) they adopt model-driven software engineering and they support all the aforementioned steps, i.e. profile definition, automated simulation code generation and b) they utilize different simulation environments.

2.1. MARTE Profile and Related Tools

MARTE is a UML profile proposed by the OMG [24] in 2009 to support 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]). Different SysML diagrams are utilized: block definition diagrams, parametric and activity diagrams. Basically they focus on integrating SysML requirements and VSL language, employed to specify them. VSL well-defined semantics enable the automated verification of corresponding SysML requirements using external tools.

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

2.2. 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 facilitated, using SysML external and internal block diagrams. The behavior of system components is described within CASSI using sequence diagrams, which may be transformed to a simulation model based on Petri-nets, executed by an external simulator. Although CASSI is based on MDA principles, existing standards and tools are not utilized, since it is built entirely on custom tools.

As described in [28], information system configurations defined using CASSI are evaluated using simulation to verify performance and availability requirements. Requirement verification is performed by the system designer within the external simulation environment, utilizing Service Level Objective (SOL) concept.

2.3. TTool Toolkit and Related Efforts

TTool Toolkit (http://ttool.telecom-paristech.fr) integrates numerous tools targeting real-time embedded system engineering. AVATAR SysML profile is one of them, 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 simulation approach, using timed-automata. The IFx toolkit3 [32] is also integrated within TTool framework for simulation purposes.

Model-driven engineering concepts are introduced in TTool toolkit components to automatically generate simulation code based on predefined libraries for the domain of embedded systems. Though, all the tools developed are proprietary to work within TTool environment, while no MOF 2.0 compatibl e meta-model is defined for the simulation or model checking

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

environments. Requirement verification can be facilitated by external tools, such as IFx toolkit3.

2.4. SysML-to-Arena Transformation Tools

In [7], manufacturing line system models are defined in SysML and transformed using ATL to be simulated using Arena simulation software. With the definition of a SysML profile, Arena-specific properties modeling 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.

2.5. SysML4Modelica Project

The SysML4Modelica profile endorsed by the OMG [8] enables the transformation of SysML models to executable Modelica simulation code. To embed simulation capabilities within SysML, ModelicaML profile is introduced [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 how to use SysML4Modelica profile for embedded systems engineering. In the proposed extensions, SysML requirement entity is enriched 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 simulator (MathModelica) through visual diagrams created during simulation. One possible limitation of this approach related to the fact that embedded system designer must be familiar with both SysML tools and MathModelica environment, since requirements are defined in SysML and verified in MathModelica.

2.6. DevSys Framework

The authors have proposed DEVSys framework for the simulation of SysML models using DEVS [9]. A DEVS SysML profile is defined 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, a 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]. 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 is presented in [21]. In this case, information system simulation 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. Additionally, performance-related attributes defined for all system components are calculated during simulation and integrated within the SysML EIS model after the completion of the simulation.

2.7. 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. Furthermore, the adaptation of model-based engineering principles and the utilization of standard languages to transform SysML-to-simulation model is clearly gaining momentum. Most of the approaches, even if defined to be general, focus on a specific system domain and provide corresponding model libraries within the simulation environment to simplify the transformation process.

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

Table 1: 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	- Performance and time requirements description and verification utilizing VSL - System behavior is described using activity and parametric diagrams	Non Specified	Medium	Non Specified	MDEReqTrace integrates requirement verification data from external tools within SysML
CASSI Tool	Information Systems	Petri-Nets	- Description of performance requirements - System behavior is described using Sequence diagrams	Non Specified	Low	Semi automated	Performed within external simulation tool
TTool Toolkit	Real-Time Embedded Systems	Y-Chart, Timed-Automata	- Requirements description using TEPE - System behavior is described using State Machine diagrams	Non Specified	Medium	Fully automated	Performed by external tools
SysML to Arena Tools	Manufacturing Line Systems	Arena	- Description of the specific domain incorporating Arena simulation-related characteristics	ATL	High	Fully automated	Not Specified
SysML4 Modelica	General (emphasis on real-time systems)	Modelica	- Description of performance requirements - System behavior under exploitation is defined as Test Cases using Modelica ML	QVT	High	Fully automated	Performed by external tools
DEVSys Framework	General (case study: Information Systems)	DEVS	- Facilitates the integration of simulation output with SysML models - System behavior is described using State Machine, Parametric and Activity diagrams	QVT	High	Fully automated	Performed within SysML models in the case study

The automated simulation code generation capabilities offered are constantly increasing, usually taking advantage of the support of model libraries. Most of the approaches attempt to tackle requirements verification as well, even though in many case this process is performed outside the SysML modeling tool.

3. Challenges and Future Directions

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) Definition of the simulation/domain specific profiles. In this process, efforts should concentrate on defining 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

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

existing approaches utilize the same SysML diagrams.

b) Transformation of 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) Utilization of the 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.

Regarding the estimation of system models behavior, SysML provides a set of diagrams for describing a single system's behavior (use case, activity, sequence, state machine). However, each diagram focuses on a different aspect of the system's behavior and the syntax of SysML does not enforce a strict combination of these aspects towards a unified executable behavioral model. On the other hand, simulation profiles for SysML focus on the semantics and structures of specific simulation frameworks, leading to solutions that cannot be applied in general. A systematic approach to assess these issues has not been proposed or adopted yet.

To this end, the details of existing simulation profiles for SysML should be examined thoroughly and processed to derive common concerns and structures. The latter should be further explored against the inherent concepts and attributes of the behavioral SysML diagrams, to conclude to a set of extensions and restrictions for SysML (i.e. a profile) that would enable the general, but conceptually precise and machine-usable definition of the behavior of systems.

Regarding requirements specification, simulation has been identified as an appropriate technique for the estimation of system models' performance. Hence, the obtained simulation results should be incorporated within the original system model and a comparison against the predefined, performance-related, requirements should be performed within the SysML modeling environment. However, many approaches perform requirements verification using external tools, due to acquaintance with them and also due to the lack of quantified requirements handling in the SysML requirements diagram.

In a similar manner as above, approaches proposing solutions for quantified requirements specifications should be examined in detail and in regard with the concepts of different SysML modeling elements (e.g. blocks, states, ports, actions). This would enable the definition of a general profile, capable of defining precise and quantified requirements. Therefore, generic and automated requirements verification within the SysML model could be enabled, once system performance estimation has been added in the model.

An overview of a generic architecture, incorporating all the aforementioned features is presented in Figure 1.

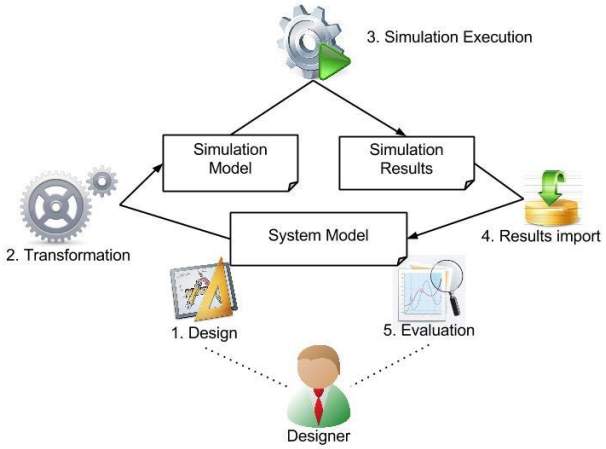

Figure 1: Proposed Architecture of SysML Model Simulation

As indicated in the figure, the main benefit of such an approach relays to the fact that the simulation environment will become agnostic for the system designer, since the designer will only interact with the SysML models through the corresponding modeling tool. In this case, the designer is not obligated to learn how to interact with the simulation

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

environment, as all simulation related activities shall be fully automated. The main restriction of the proposed approach is that it would be domain and simulator specific, in order to efficiently support automated simulation code generation. Though, in fact all of existing approaches obtain these characteristics, so it is not considered as a significant obstacle.

Towards this direction, the authors indent to develop a simulator and domain agnostic MBSE methodology to perform SysML model simulation based on the proposed architecture and construct an integration framework for different tools and approaches.

4. Conclusions and Future Work

SysML became a valuable tool for system-of-systems modeling establishing a growing community of system designers using it. As simulation execution features are not integrated within SysML there is a need for transforming SysML models to simulation models. This effort is hindered by two factors: a) the lack of standardization in the automated simulation code generation process and b) the support of simulator-specific SysML profiles to enable the automated generation of simulation code.

Taking into account the approaches presented in the paper and the vision most of them promise to implement, we proposed a generic architecture, based on MBSE principles, for automating the simulation code generation process. Many approaches, as SysML to Arena Tools, SysML4 Modelica and DevsSys framework, are compatible with the proposed architecture.

The development of a simulator-agnostic framework to support SysML model simulation remains a challenge. The existence of a generic discrete-event and continuous simulation XMI meta-model is the first step towards this direction. Each of them may be initialized, utilizing the transformation of different SysML diagrams. Such a meta-model may also serve as an integration framework for different simulation tools and approaches.

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, in: Proceedings of 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: Proceedings of 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, Simulating SysML models: Overview and challenges, in: Proceedings of IEEE 10th Conference on SoSE, San Antonio, Texas, May 2015.

[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: Proceedings of Aerospace Conference, 2011 IEEE, 2011, pp. 1–15.

[17] 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.

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.

[20] 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] A Tsadimas, GD Kapos, V Dalakas, M Nikolaidou, D Anagnostopoulos, Integrating Simulation Capabilities into SysML for Enterprise Information System Design, in: Proceedings of IEEE 9th Conference on SoSE, Adelaide, Australia, June 2014.
[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: Proceedings of 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, in: Proceedings of 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, in: Proceedings of 6th International Conference on 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: Proceedings of NOTERE 2011, 11th Annual International Conference on, 2011, pp. 1–10.
[32] 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: in: Proceedings of Systems Conference (SysCon), 2014 8th Annual IEEE, IEEE, 2014, pp. 531–538.

Mara Nikolaidou is a Professor in the Department of Informatics and Telematics at Harokopio University of Athens. Her research interests include SoS engineering, system modeling and simulation, service-oriented architectures and BPM. Over the last years she actively participated in numerous projects on system engineering, service-oriented architectures, digital libraries and e-government. She is currently participating in research project focusing on SoS engineering, Internet of Things and Smart Cities. She has published more than 100 papers in international journals and conferences.

George-Dimitrios Kapos is currently performing research for his PhD thesis at the Department of Informatics and Telematics, Harokopio University of Athens, Greece. In parallel, he works as an analyst and software developer at the IT Department of the Greek Consignment Deposit & Loans Fund. He obtained his BSc in Informatics and an MSc degree in Advanced Information Systems, both with honors from Informatics & Telecommunications Department, National Kapodistrian University of Athens, Greece. His research interests include model-based systems engineering, simulation and meta-modelling.

Anargyros Tsadimas since 2004 is working at the Harokopio University as a research associate where is currently Ph.D. Candidate at the Department of Informatics & Telematics. He received his B.Sc. in Applied Informatics from the University of Macedonia in 2002 and his MSc in Advanced Information Systems from the Department of Informatics & Telecommunications of the National and Kapodistrian University of Athens in 2005. His research interests lie in the field of Systems Engineering and Modeling. He has several publications in international conference proceedings and he has been participated in numerous R&D projects.

Vassilis Dalakas obtained a BSc in Physics, a MSc degree (honors) in digital signal processing, and a PhD degree in digital communications, all from the University of Athens (UoA), Greece, in 1998, 2002, and 2010, respectively. Since 2001, he has been affiliated with the Harokopio University of Athens (HUA), Greece, as a Research Fellow (since 2001) and as a network and system administrator since 2005. His research interests include wireless digital communications, as well as modeling and simulation standardization methods. In these areas, he has co-authored several papers and two book chapters.

Dimosthenis Anagnostopoulos is a Professor in the Department of Informatics and Telematics at Harokopio University of Athens. He also currently serves as the Rector of Harokopio Univerisity of Athens and Head of the Department. He has published more than 100 papers in international journals and conferences. His research interests include discrete event simulation, faster-than-real-time simulation, modeling and simulation of distributed information systems. He has actively participated in numerous projects related to simulation, e-government and information systems.

Copyright (c) 2016 Advances in Computer Science: an International Journal. All Rights Reserved.