Moving Integrated Product Development to Service Clouds in the Global Economy
J. Cha et al. (Eds.)

doi:10.3233/978-1-61499-440-4-617

SysML-based Model Driven Discrete-Event Simulation

Yitao LIU[a, b, 1], Prashanth IRUDAYARAJ[a, c], Feng ZHOU[a], Roger J. JIAO[a, b] and Joseph N. GOODMAN[c]

[a] *The G.W. Woodruff School of Mechanical Engineering,*
[b] *Renewable Bioproducts Institute,*
Georgia Institute of Technology, Atlanta, Georgia 30332, USA
[c] *Georgia Tech Research Institute, Atlanta, Georgia 30318, USA*

Abstract. The analysis of modern complex systems is becoming more dependent on simulation approaches. The variety of discrete-event simulation tools and their complex nature have caused problems of low efficiency in such simulation processes. The quality of simulation also largely depends on the analysts understanding to the whole system. These short comings are holding the discrete-event simulation back form wider utilization. A Model Driven Architecture approach is employed in this paper to support the practice of discrete-event simulation modeling based on SysML. A case study of photovoltaic solar power system lifecycle cost analysis is conducted as a validation of the proposed method.

Keywords. Model driven, discrete-event simulation, SysML, system modeling.

Introduction

The performance of a complex system is always hard to analyze by analytical approaches in engineering practice. Discrete-event simulation (DES) tools are used to assess such problems by modeling a system as a discrete sequence of events in time. Such model is conceptually built using *ad-hoc* methods by the simulation analysts based on their understanding of the system. These *ad-hoc* methods have no standards or structure to follow and can take various forms such as documents, diagrams, databases, etc.

Using simulation language to represent the system impacts the fidelity of the communication between the domain engineers which may introduce doubt as to whether the simulation analysts have grasped fully their intent. In order to ensure that the simulation analysts receive the right information, significant time and effort are expended in this phase of any simulation project. Moreover, the informality of these methods hinders the re-usability of the system descriptions or investigating automatic model transformations [1].

A more generalized language for system modeling which enables domain engineers with different background knowledge accessing is needed in the practice of

[1] Corresponding Author: Yitao Liu, Georgia Institute of Technology, Atlanta, Georgia 30332, USA; E-mail: yliu410@gatech.edu.

DES. Such language can bridge the gap between domain engineers and simulation analysts and also should serves as a multi- disciplinary modeling tool which will allow cooperation of domain engineers of different sub-systems.

Technical challenges: They can be summarized as the following three points:

- Semantic entanglements in traditional simulation modeling processes may lead to inefficiency, and accuracy of the model depends on the understanding of simulation analysts;
- A conceptual representation of the system which is understandable to all domain engineers is needed, which should be systematical and coherent;
- The transformation from the conceptual representation to domain simulation language is the key issue in this approach; only a reliable and efficient transformation will show its significance.

Problem solution: The System Modeling Language (SysML) is a general purpose modeling language for systems engineering applications and its scope goes through a wide range of systems, or systems of systems, including hardware, software, processes, facilities, etc. It is an extension of UML designed to support systems engineering in general. As software generation from UML models is a common practice, it is reasonable to use SysML, which can be considered as a system modeling extension of UML, for complex system modeling and simulation.

SysML is a graphical language which is easier to understand, in terms of its description of the system being simulated, than various DES tools. It is based on the eXtensible Markup Language (XML) standard, which means it is object oriented and easily transformed "automatically" into its corresponding simulation model. By using SysML as an intermediate representation of the DES model, domain engineers can get involved into the modeling process and thus guarantee a coherent understanding in modeling. A simplified framework comparison is shown in Figure 1.

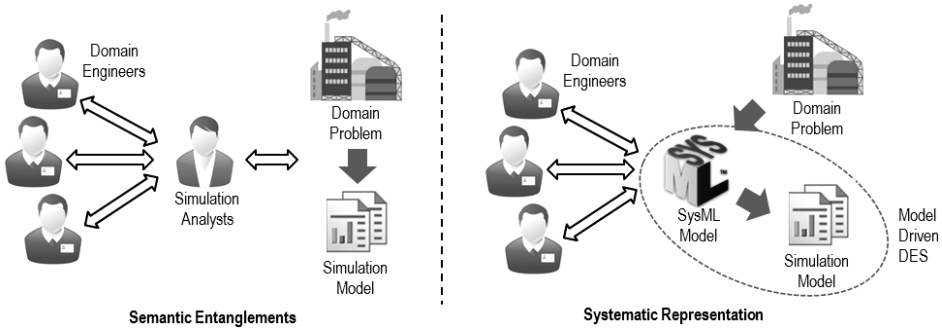

Figure 1. Frameworks of traditional modeling process and model driven approach.

Related work: SysML is widely used in system modeling since its introduction in 2006. Later in 2007, the first trial of SysML model transformation to DES is done by Huang *et al* [2]. They successfully converted a standard three-unit machine shop model into Em-Plant. Then in 2009, McGinnis *et al* [3] developed a SysML driven Arena model of the same machine shop example. A further study by this group is summarized in 2012, in which the Atlas Transformation Language in Eclipse is used to do the transformation work. SysML model driven technique is also used in other research domains such as system-on-chip development [4, 5], software engineering [6, 7],

mechatronic system design [8, 9] and requirement engineering [10]. Although the SysML model driven design and simulation has been studied for seven years, there are still not many papers on practical DES applications.

In this paper, a SysML model driven DES framework is introduced with a case study of solar power system lifecycle modeling. The following paper proceeds with introduction of SysML modeling language, Arena simulation language and their transformation from Section 1 to Section 3. Then the case study is demonstrated in Section 4 and concluded in Section 5.

1. SysML Model

Recently, SysML has emerged as the preferred method for modeling complex systems. It has also been considered for use in the conceptual design phase in a systematic product design process. It is proposed to describe the system more clearly and reduce the effort for any further modification to the system model. It can not only be regarded as a common language between domain engineers, but also a source code which can be automatically transformed into Arena simulation language. While it is a formal language, conforming to Meta-Object Facility, it has a graphical user interface, making most diagrams relatively easy to understand. The block is the basic unit of presentation in SysML and can be used to represent hardware, software, facilities, personnel, or any other system element. Typically, there are four kinds of SysML diagrams, i.e., structure diagram, behavior diagram, requirements diagram, and parametric diagram as shown in Figure 2.

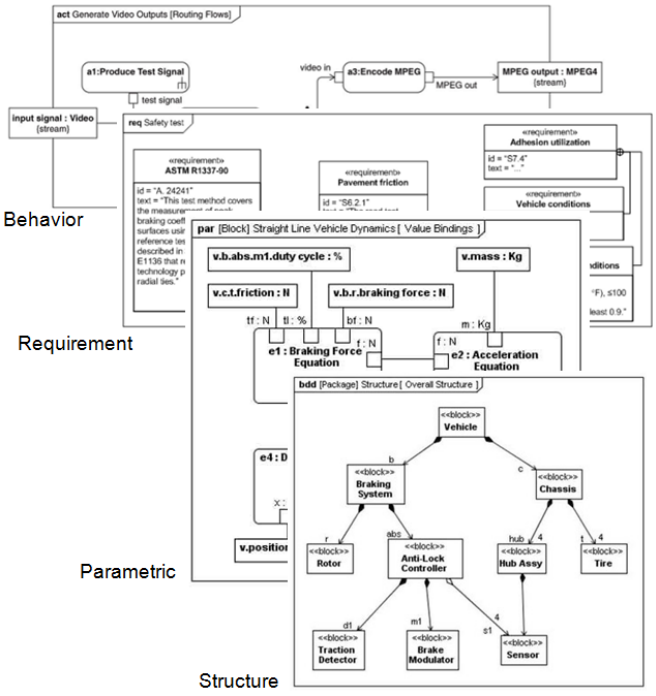

Figure 2. The four pillars of SysML.

The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram. The use-case diagram provides a high-level description of functionality that is achieved through interaction among systems or system parts. The activity diagram represents the flow of data and control between activities. A sequence diagram represents the interaction between collaborating parts of a system. The state machine diagram describes the state transitions and actions that a system or its parts perform in response to events. Arena is a process-oriented modeling tool for discrete-event systems. In other words, the modeling in Arena environment is structured as a workflow of stepwise activities and actions. Therefore, the activity diagram in SysML is the appropriate tool to develop the Arena stereotypes, using activity diagram components' meta-classes.

2. Arena Simulation Model

In Arena, modules are basic elements that represent processes or logic. Connector lines are used to join these modules together and specify the flow of entities. While modules have specific actions relative to entities, flow, and timing, the precise representation of each module and entity relative to real-life objects is subject to the analysts. Arena is a process-oriented modeling tool for discrete-event systems. In other words, the modeling in Arena environment is structured as a workflow of stepwise activities and actions.

Other than building the simulation models by GUI operation, Arena can be integrated with Microsoft technologies, including Visual Basic, Access database and ActiveX controls. Through such interface, DES modeling transformation and automation become possible. Arena simulation model is run in sequence of events, and the events can be standard modules of Arena or VBA custom codes. A basic sequence of events in an Arena model is shown in Figure 3.

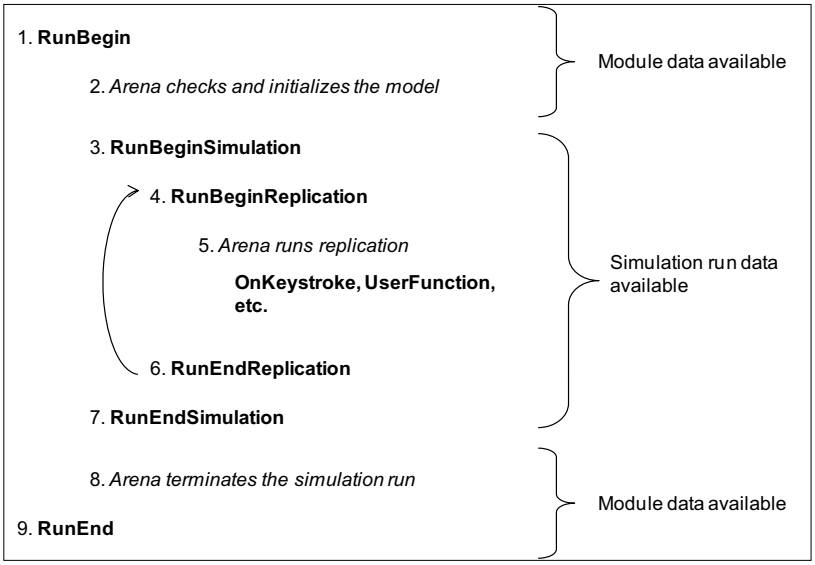

Figure 3. Arena/VBA sequence of events.

The basic building blocks are elements, such as entities, queues, resources and sequences, and the process blocks which affect them, some of which are shown in Figure 4.

Create 1	Push (possibly) batches of entities into the model with a (possibly) random time between.	Assign 1	Assign values (especially Attributes) when an entity passes through.
Process 1	Models Queue-Seize-Delay-Release of Resource, or any part of this (like pure Delay).	Record 1	Record information when entities pass through, typically statistics on entities.
Decide 1	Make decisions about where to go next based on conditions or chance.	Batch 1	Combine multiple entities into a single entity.
Dispose 1	Take entities out of the model and (perhaps) record statistics.	Separate 1	Split multiple entities that were combined, or duplicate a single entity.

Figure 4. Basic process modules in Arena.

The main task of model transformation is to set the rules in transformation codes to convert SysML activities into Arena elements and modules.

The other important function of Arena is the data input analyzer. A valid DES model will not work properly without a valid input data. Normally the data collected from real world are limited, input data fitting is then necessary. Such data is commonly governed by certain kinds of distributions and input analyzer is used to pre-process the raw input data to find a marching distribution property. This data pre-process procedure also should be automated and become one part of the SysML driven DES process.

3. Model Transformation

The model transformation process allows simulation analyst to build this logic in SysML using Arena semantics. Such SysML model is more readable for domain engineers with different background and thus enables collaboration on DES modeling process.

The input files of model transformation are XML files of SysML models and text based data files. The XML files contain SysML action diagrams which have all modeling information of Arena events. As the common composition of an arena event is known, we can build an Arena processes and elements template which stores all the information needed. Then the workload of building SysML model will be reduced as most of the standard processes and elements can be directly picked from template and no further definition is needed. Some basic element templates are shown in Figure 5.

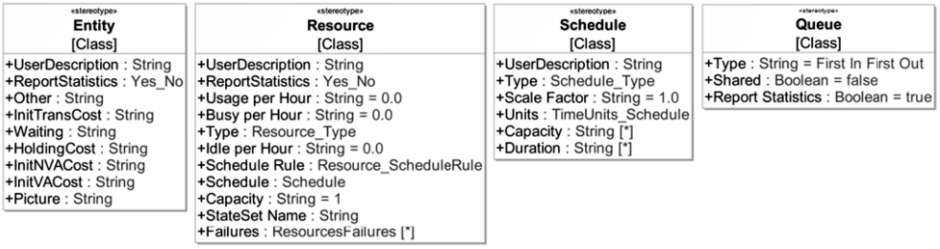

Figure 5. Part of basic element template.

Data input analyzer is a plug-in of Arena and its data fitting function relies on user manual inputs. A third party data fitting program using dfittool of Matlab is used in order to gain the automatic data conversion and distribution parameter generation.

The overall model transformation process framework is shown in Figure 6.

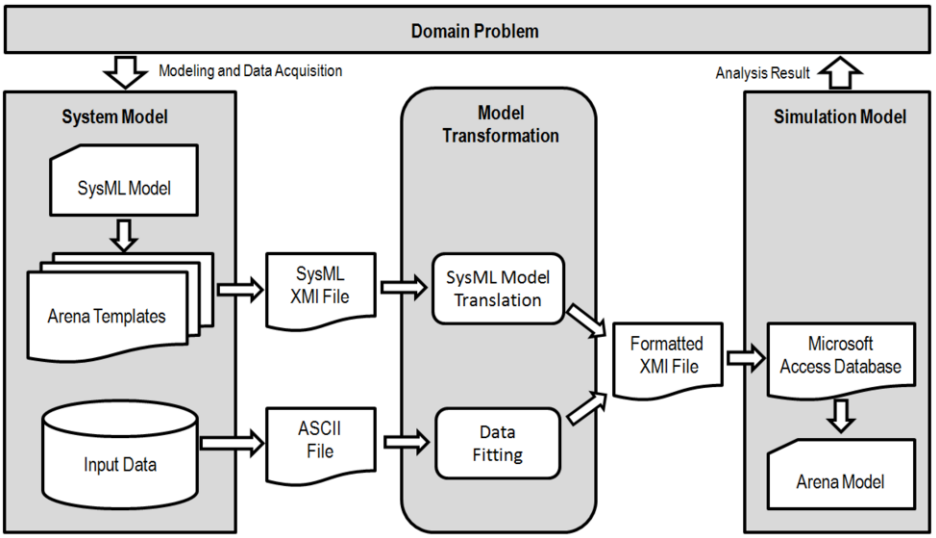

Figure 6. The model transformation framework.

The XMI files in the transformation process will be converted several times in a Java environment. Input data will also be structured in a proper format that enables reading through the Matlab package. The data will be fitted and a parameter set will be generated and saved to the formatted XMI files. Microsoft Access Database (MDB) files serves as an intermediate to the XMI file and Arena DOE format model. All the transformation can be packed into one executable program and operated by simulation analysts.

4. Case Study

A solar power system lifecycle including manufacturing, installation and maintenance is modeled and simulated in this case study. A final result of its lifecycle human labor is calculated from the simulation.

4.1. Solar Power System Lifecycle Cost Analysis

The current solar power systems are either directly using photovoltaics (PV), or indirectly using concentrated solar power (CSP). The solar power system studied in this research is PV, which typically consists of photovoltaic modules for power production, racking systems for module mounting, inverters for power conversion and batteries for storage. The configuration of the system is modified to meet the power generation needs of specific applications. The affordability of solar power systems in general has increased in recent years because of advances in cell efficiency and manufacturing technology. This has primarily reduced the cost of the modules themselves, but there

are still potential ways to reduce costs in the balance of systems. The labor component of the balance of system is a function of the system design and various activities that support its implementation in a given scenario. This creates the complexities in evaluating the overall lifecycle labor cost of the system. As a result, it is often difficult to compare the relative labor cost of solar power systems to support design or product selection.

Discrete event simulation provides an opportunity to compare the labor costs of different solar power systems. And a model driven DES have more flexibility and is easier to be understood by the decision makers while balancing the lifecycle design. The solar system used in this paper is shown in Figure 7.

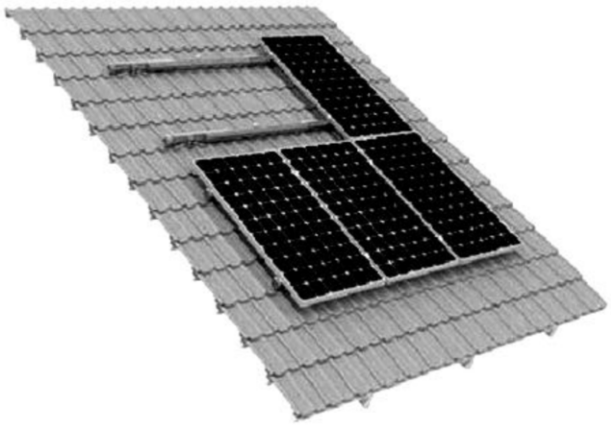

Figure 7. Roof solar panel system.

4.2. Data Collection and Fitting

Data for the three phases can be collected through various means. For the purposes of this case study, distributions of processing times for the manufacturing activities were assumed based on the historical data due to the fact that most of these activities can be well controlled in the modern factories. The data collection procedure is described in detail by Goodman et al [11]. A sample data fitting result is shown in Table 1.

Table 1. Input data fitting result for solar panel installation.

	Coordinate Racking Module	**Unload Racking Module***	**Prepare Fastener Module**	**Prepare Base**
Fitted Distribution	Lognormal	DISC	Lognormal	Lognormal
Estimated Parameters	log(mu):-2.9835; log(sigma): 0.7563	-	log(mu):-5.9212; log(sigma): 0.9322	log(mu): -5.8601; log(sigma): 0.6174
Arena Parameters	log(mean):.0674; log(std): .0035	(0.5,0.045,1,0.022)	log(mean):.0041; log(std): .0000237	log(mean):.0034; log(std): .0000055
Log Likelihood	55.8145	-	91.9493	158.05
Chi-square goodness-of-fit test	Pass	-	-	Pass
K-S goodness-of-fit test	Pass	-	Pass	Pass

4.3. SysML Model

Activity diagrams for all solar system lifecycle is modeled in SysML, they are simplified and can be linked to the Arena templates to generate a full size XML model file. A SysML model of solar panel manufacturing process is shown in Figure 8.

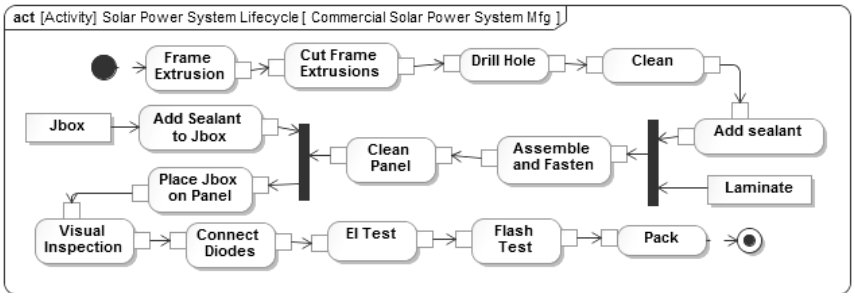

Figure 8. The activity diagram of commercial solar power system manufacturing.

4.4. Transformed Arena Model

The transformed models have all the features of the SysML model. Although the block layout and simulation parameters still need further modifications, the Arena model is runnable and showing reasonable response. The models are also manually modified in assign block to insure the right parallel sequence. The Arena model same process displayed in section 4.3 is shown in Figure 9.

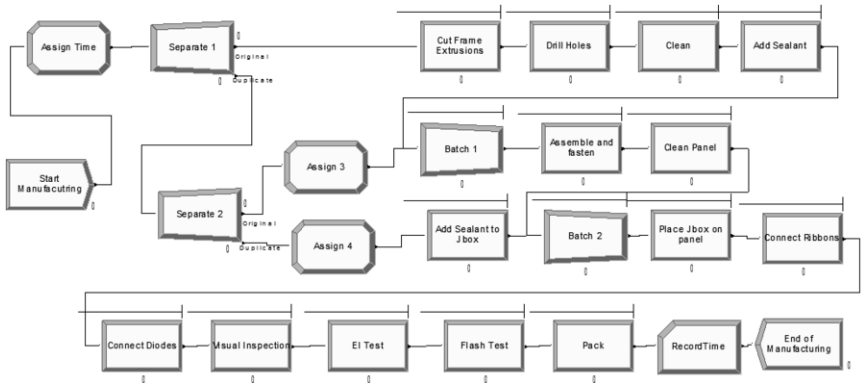

Figure 9. The transformed arena model of commercial solar power system manufacturing.

Based on the simulation results, we can tell the seconds/watts used in manufacturing, installation and maintenance of the given solar power system. Substantial reduction in seconds/watts can be obtained in the modification of SysML models which cause the change of the system. A simulation output comparison of a conventional solar power system lifecycle cost with its modification is shown in Figure 10.

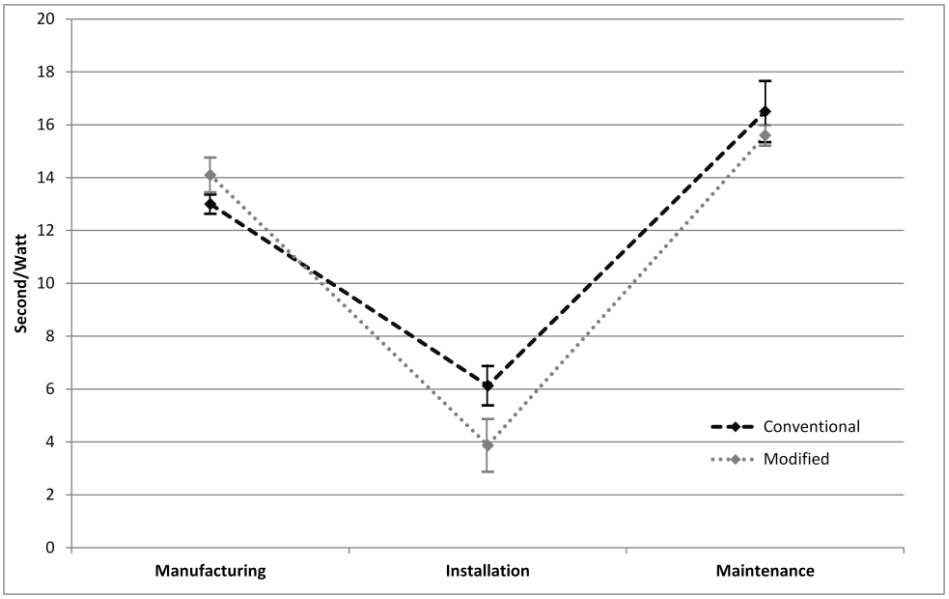

Figure 10. The simulation result of a solar power system lifecycle cost before and after modification.

5. Conclusions

The SysML based model driven approach is a framework for modeling discrete-event simulation of complex systems. It can be applied to various kinds of systems and simplifies the modeling effort. SysML is simpler than the domain-specific languages such as Arena. Hence, such advantage allows domain engineers from all sub-systems to understand the simulation model and make necessary changes. The separation of specific SysML model and Arena template can reduce the repeated modeling work. Any update or modification of the discrete-event simulation tools can be caught by modifying the template. It could also reduce the effort of changing discrete-event simulation tools if models are built properly.

Through the case study, we can determine lifecycle labor costs for given solar systems. It can be used to identify relative costs for solar power systems at various stages of decision making. At the conceptual design stage, it can help domain engineers evaluate the relative costs of competing concepts. The model driven approach also can reduce the effort in changing the models for "what if" analysis. Comparison can also be made in different revisions.

The current proposed approach still have limitations on the transformation of the two models. Some details in Arena models are not yet covered by the SysML model and need manual modification after transformation. The Arena advanced process blocks are still not covered in current templates, which can be added in the future. An inversed transformation from the DES model to SysML can also be a future consideration as it may helps in cases that domain model already exists.

Acknowledgements

This research is partially supported by the Department of Energy under the SunShot award program for Extreme Balance of System Hardware Cost Reductions (Award No. DE-EE0005441.001).

References

[1] O. Batarseh & L.F. McGinnis, System modeling in SysML and system analysis in Arena, *In Proceedings of the Winter Simulation Conference*, (2012), 258.
[2] E. Huang, R. Ramamurthy & L.F. McGinnis, System and simulation modeling using SysML, *In Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, (2007), 796-803.
[3] L. McGinnis & V. Ustun, A simple example of SysML-driven simulation, *In Simulation Conference (WSC), Proceedings of the 2009 winter*, (2009), 1703-1710.
[4] E. Riccobene & P. Scandurra, Integrating the SysML and the SystemC-UML profiles in a model-driven embedded system design flow, *Design automation for embedded systems*, **16** (2012), 53-91.
[5] M. Bombino & P. Scandurra, A model-driven co-simulation environment for heterogeneous systems, *International Journal on Software Tools for Technology Transfer*, **15** (2013), 363-374.
[6] P. Weyprecht & O. Rose, Model-driven Development of Simulation Solution based on SysML starting with the Simulation Core, *In Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, (2011), 189-192.
[7] C. Shih, & B.C. Liang, A model driven software framework for Zigbee based energy saving systems, *In Intelligent Systems, Modelling and Simulation (ISMS)*, (2012), 487-492.
[8] Y. Liu, W. Yuan, H. Fan, & Y. Cao, Research on information integration framework of SysML based model driven design of complex products, *China Mechanical Engineering*, **23** (2012), 1438-1445.
[9] V.H. Ngo & T. Soriano, A model transformation process to realize controllers of ship autopilot systems by the specialized MDA's features with UML/SysML, In Mechatronics (MECATRONICS), 2012 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM), (2012), 20-26.
[10] M.D.S. Soares & J. Vrancken, Model-Driven User Requirements Specification using SysML, *Journal of Software*, **3** (2008).
[11] J. Goodman, K. Nagel, M. Wren, and J. Morris, Applying lean process principles to improve labor efficiency of solar photovoltaic installations, *presented at the Construction Industry Research Conference*, (2013).