MATLAB/Simulink's Variant Manager vs SESToPy

Christina Deatcu*, Thorsten Pawletta, Hendrik Folkerts

Hochschule Wismar - University of Applied Sciences, Research Group CEA, Philipp-Müller-Straße 14, 23966 Wismar, Germany; *christina.deatcu@hs-wismar.de

SNE 29(1), 2019, 39-43, DOI: 10.11128/sne.29.sw.10466
Received: March 11, 2019 (Selected ASIM GMMS/STS 2019 Conference Publication); Accepted; March 20, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper describes how a complex case study for variability modeling and simulation from the documentation of MATLAB/Simulink can be remodeled with the extended System Entity Structure and Model Base (eSES/MB) approach using the Phython-based tool *SESToPy* and the accompanying modelbuilder *SESMoPy*.

Introduction

Generally, variability modeling can be seen as an approach to describe more than one system configuration. According to Capilla and Bosch [1], a software variability model has to describe the commonality and variability of a system at all stages of the software lifecycle. In simulation engineering, the problem of variability modeling is well known from the eighties. One of the first high level approaches for variability modeling in the design phase was introduced with the System Entity Structure (SES) by Zeigler in 1984 [2] and is constantly enhanced until today [3] [4]. The SES is a high level approach for variability modeling, particularly in simulation engineering. An SES describes a set of system configurations, i.e. different system structures and parameter settings of system components. In combination with a Model Base (MB), executable models can be generated from an SES.

A common tool for today's engineering applications is MATLAB/Simulink. It offers pragmatic solutions for variability modeling and can be seen as a quasi-standard in engineering. In the following section, the example of a power window control system modeled with different degrees of detail is introduced. This application is taken from MATLAB/Simulink's examples section and can be found in the online documentation [5]. Therefore, the Simulink model did not have to be created but needed to be analyzed. Remodeling the example using the extended System Entity Structure and Model Base (eSES/MB) approach as described by Pawletta et al. [6] was successfully done.

After the problem description in Section 1, Sections 2 and 3 describe the two modeling approaches using the case study. Finally, a comparative evaluation is tried regarding: (i) the modeling effort, (ii) the clarity, (iii) the reusability, and (iv) the maintainability.

1 Problem Description

The passenger-side power window system of an automobile is modeled and simulated. The power window can be controlled from both the driver's and the passenger's side. Furthermore, closing should be stopped for security reasons, in case an obstacle is detected during upward movement of the window. The window shall be lowered by some centimeters in this case.

The system is modeled with different degrees of detail and with using different modeling concepts. From this problem specification, multiple model structures and configurations result, which are called variants. For all variants, main model parts are two switches for controlling the window, the control model, a process model of the window, and a model for 3D-animation. The occurrence of an obstacle can be controlled interactively. Furthermore, some outputs are needed to observe the window's behavior. A complete description of the *Power Window Control Project* example can be found in MATLAB's online documentation [5].

2 Implementation with Simulink's Variant Manager

Simulink as one of the most popular tools for model-based development provides special blocks for switching between model structures, the *Variant Subsystems Blocks*. Project management is facilitated by the *Simulink Project* capabilities. Anyhow, that means that all variants need to be coded in only one model. Models of this kind are called 150%-models. The main Simulink model is depicted in Figure 1. The model comprises five variant subsystems: (i) the *driver_switch*, (ii) the *passenger_switch*, (iii) the model for obstacle detection, which is a submodel of the *power_window_control_system* model, (iv) the *window_system* model, and (v) the model *window_world*, that offers optional 3D-animation.

Both switches can operate either in a mode called normal mode or use a communication protocol (CP) implementation. For obstacle detection (DOE) four variants are available. The first variant is a simplistic continuous system model (Cont), the second uses power effects (PE), the third works with a visualization (Vis), and the last provides support for realistic armature and the communication protocol (RA CP). The *window_system* model subsystem (WS) comprises three variants, one simple continuous model (Cont), one reproducing power effects and additional 3D-visualization options (PE Vis), and a third variant where the realistic armature and the communication protocol is included (RA CP). The two variants for the *window_world* are a Simulink 3D animation or no animation at all.

Some of the variants use Stateflow, which is The MathWorks' implementation of state machines, and/or physical modeling, i.e. Simscape, submodels. One important aspect when modeling the variants is, that the number and names of input and output ports of variant subsystems need to be the same, no matter which variant is chosen.

The active variant of the main model can be programmatically changed prior to simulation via the control variable *CV*. Table 1 lists the possible configurations and corresponding values of the control variable *CV*. Not all combinations of selected variants in the subsystems are valid. If e.g. for the switches the communication protocol variant is chosen, the active variants of *window_system* and of the obstacle detection modeled in *power_window_control_system* need to be the communication protocol implementations, too. The variation in *window_world* is not addressed in Table 1, because all configuration sample scripts in the MATLAB/Simulink example implementation use the *Simulink_3D_Animation View* variant, none uses the *No_View* submodel.

	Switches	**DOE**	**WS**
CV=1	Normal	Cont	Cont
CV=2	Normal	PE	PE Vis
CV=3	Normal	Vis	PE Vis
CV=4	Normal	RA CP	RA CP
CV=5	CP	RA CP	RA CP

Table 1: Valid variants, control variable values and selected submodels.

To allow only valid variant combinations, the value of *CV* is evaluated prior to simulation and mapped to specific control variables, called *variant control*, associated with the single variant subsystems. A variant choice is active, when the associated variant control evaluates to TRUE.

3 Implementation with the eSES/MB Approach

For remodeling of the power window example, the platform-independent and open source variability modeling tools *SESToPy* [7] and *SESMoPy* were used. These Python-based tools are developed by the research group Computational Engineering and Automation (CEA) at Hochschule Wismar. Both tools and the infrastructure they are used with are described in detail in [8].

A family of systems, which in this context means all possible variants, can be defined within a System Entity Structure (SES) using SESToPy. An SES is represented by a directed acyclic graph with an amount of entity nodes, descriptive nodes and attributes. For usage with model generation, entity nodes are linked to basic models organized in a Model Base (MB). Attributes of an entity node correspond to the parameters of the belonging model component. The available three kinds of descriptive nodes specify the relationship between entities. Aspect nodes are used to define the composition of entities, multi-aspect nodes are a special kind of as-

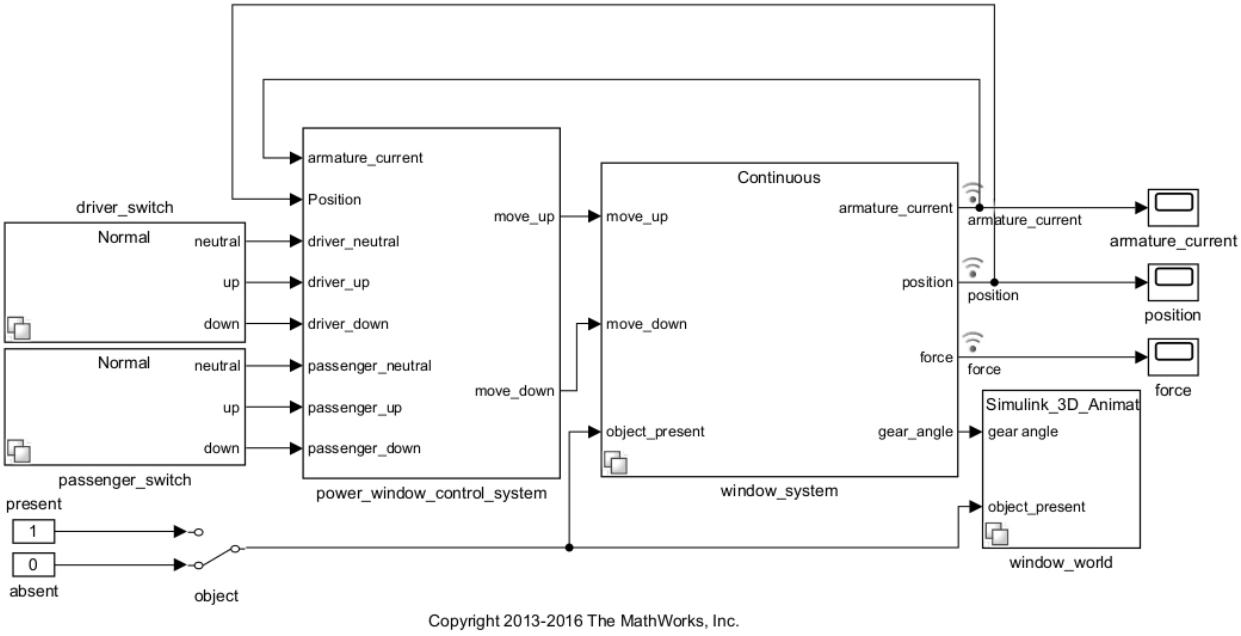


Figure 1: Overall Simulink model structure according to The MathWorks [5].

pect nodes, where all component entities are of the same type. The third descriptive node type, the specialization node, describes the taxonomy of an entity.

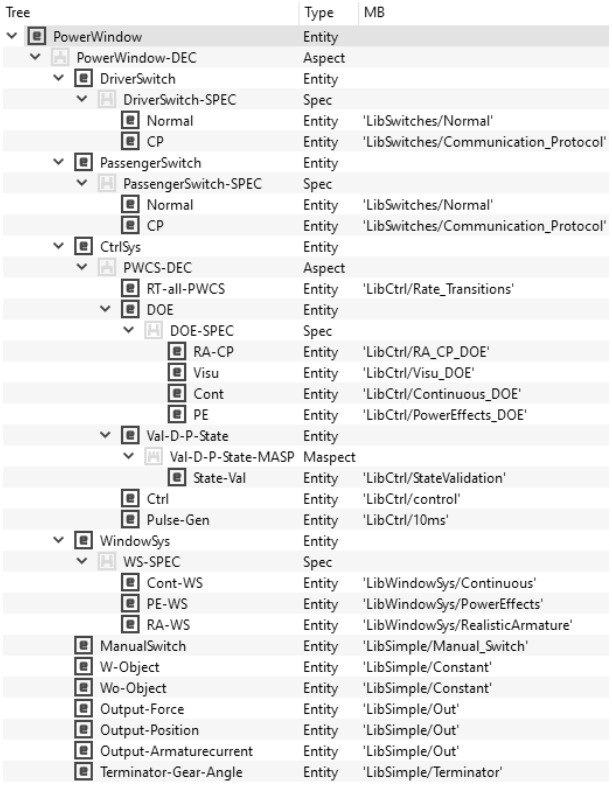
Descriptive nodes can be seen as variation points of an SES. Associated with the descriptive nodes, there are rules which need to be evaluated when deriving one specific system variant. Each specific system variant comprises of a system structure and a parameter configuration.

The process of deriving a system variant from an SES is called *pruning* and the result is a Pruned Entity Structure (PES). The PES still is a directed acyclic graph like the SES, but without any variation points. Applying the *flattening* method to a PES, all inner nodes are removed to obtain a Flattened Pruned Entity Structure (FPES). In conjunction with an MB, a fully configured and executable model can be generated from an FPES using the modelbuilder tool SESMoPy.

First step for remodeling the power window example was to identify the basic models which then needed to be organized in an MB. As the MB for the implementation of this example application the basic models were structured in four libraries. In the libraries, preconfigured blocks or submodels can be stored to reduce parametrization effort when defining the SES.

The library *LibSimple* contains basic models which are blocks copied directly from Simulink's block library to allow preconfiguration and to make the MB independent from Simulink versions. The other three libraries *LibSwitches*, *LibCtrl*, and *LibWindowSys* contain more complex models and can be seen as user-defined libraries. Then all configuration variants were coded in an SES tree according to the structure of The MathWorks' 150%-model. Figure 2 shows the SES tree modeled in SESToPy.

The entire example and all variants are covered except the 3D-animation variation in *window_world*. The SES was developed step by step making use of SESToPy's capability to combine several SES trees with the *merge* method [8]. The variation points are expressed by the specialization nodes *DriverSwitch-SPEC*, *PassengerSwitch-SPEC*, *DOE-SPEC*, and *WS-SPEC*. For each specialization node, a specialization rule is defined. During pruning the rules are evaluated and it is decided, which of the child nodes will be part of resulting PES. How models are connected is defined with the coupling attribute at aspect and multi-aspect nodes. Since couplings can be set dynamically here, names and number of input and output ports are variable.



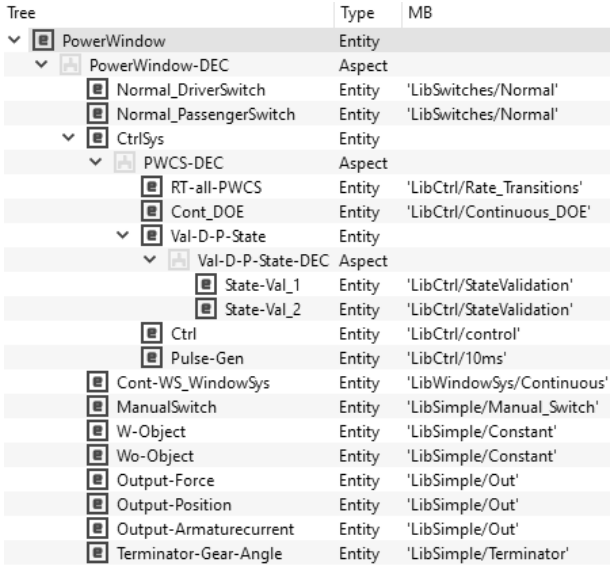
Tree	Type	MB
PowerWindow	Entity	
PowerWindow-DEC	Aspect	
DriverSwitch	Entity	
DriverSwitch-SPEC	Spec	
Normal	Entity	'LibSwitches/Normal'
CP	Entity	'LibSwitches/Communication_Protocol'
PassengerSwitch	Entity	
PassengerSwitch-SPEC	Spec	
Normal	Entity	'LibSwitches/Normal'
CP	Entity	'LibSwitches/Communication_Protocol'
CtrlSys	Entity	
PWCS-DEC	Aspect	
RT-all-PWCS	Entity	'LibCtrl/Rate_Transitions'
DOE	Entity	
DOE-SPEC	Spec	
RA-CP	Entity	'LibCtrl/RA_CP_DOE'
Visu	Entity	'LibCtrl/Visu_DOE'
Cont	Entity	'LibCtrl/Continuous_DOE'
PE	Entity	'LibCtrl/PowerEffects_DOE'
Val-D-P-State	Entity	
Val-D-P-State-MASP	Maspect	
State-Val	Entity	'LibCtrl/StateValidation'
Ctrl	Entity	'LibCtrl/control'
Pulse-Gen	Entity	'LibCtrl/10ms'
WindowSys	Entity	
WS-SPEC	Spec	
Cont-WS	Entity	'LibWindowSys/Continuous'
PE-WS	Entity	'LibWindowSys/PowerEffects'
RA-WS	Entity	'LibWindowSys/RealisticArmature'
ManualSwitch	Entity	'LibSimple/Manual_Switch'
W-Object	Entity	'LibSimple/Constant'
Wo-Object	Entity	'LibSimple/Constant'
Output-Force	Entity	'LibSimple/Out'
Output-Position	Entity	'LibSimple/Out'
Output-Armaturecurrent	Entity	'LibSimple/Out'
Terminator-Gear-Angle	Entity	'LibSimple/Terminator'

Figure 2: SES tree of the power window control system family in SESToPy.

In analogy to the control variable *CV* and the variant control variables in The MathWorks' 150%-model, three SES variables are used. The SES variables are *driver_MODE*, *detect_O_E*, and *window_system*. Ranges of the SES variables and combinations among them are restricted by defining semantic conditions. Thus, only valid variants can be generated, i.e. the SES can be pruned only to a valid PES. Figure 3 shows the example of one possible PES. This PES corresponds to the Simulink variant, where *CV*=1. After flattening, model generation from the resulting FPES was finally successfully done with SESMoPy.

4 Conclusion

Both approaches offer the possibility to model and simulate variability systems. A significant difference is that with the Variant Manager interfaces of submodels are static, while the eSES/MB approach allows to define variable interfaces and couplings. Regarding the modeling effort the approaches do not differ considerably, but modeling with Simulink's Variant Manager can be seen as a bottom-up procedure while modeling with the eSES/MB approach is rather top-down. The eSES/MB approach gives more clarity during the modeling process, because the overall structure of the model can be captured with one sight. If one uses Simulink's *Model Explorer* to determine the structure, one cannot see, that and where a model contains variant subsystems until one selects a variant subsystem block. The *Variant Manager* offers a view, where the overall structure is displayed, but e.g. block properties cannot be seen then. Information about the model is distributed over several tools, which may confuse new users. A survey among our students came to the result, that eSES/MB is a lot easier to get started with. Reusability of models is ensured for both approaches but may differ in the effort. The maintainability is closely associated with the reusability. Admittedly, a final comparison is not possible on the basis of just one example. According to the available findings, the eSES/MB approach appears to be easier in use for beginners.



Tree	Type	MB
PowerWindow	Entity	
PowerWindow-DEC	Aspect	
Normal_DriverSwitch	Entity	'LibSwitches/Normal'
Normal_PassengerSwitch	Entity	'LibSwitches/Normal'
CtrlSys	Entity	
PWCS-DEC	Aspect	
RT-all-PWCS	Entity	'LibCtrl/Rate_Transitions'
Cont_DOE	Entity	'LibCtrl/Continuous_DOE'
Val-D-P-State	Entity	
Val-D-P-State-DEC	Aspect	
State-Val_1	Entity	'LibCtrl/StateValidation'
State-Val_2	Entity	'LibCtrl/StateValidation'
Ctrl	Entity	'LibCtrl/control'
Pulse-Gen	Entity	'LibCtrl/10ms'
Cont-WS_WindowSys	Entity	'LibWindowSys/Continuous'
ManualSwitch	Entity	'LibSimple/Manual_Switch'
W-Object	Entity	'LibSimple/Constant'
Wo-Object	Entity	'LibSimple/Constant'
Output-Force	Entity	'LibSimple/Out'
Output-Position	Entity	'LibSimple/Out'
Output-Armaturecurrent	Entity	'LibSimple/Out'
Terminator-Gear-Angle	Entity	'LibSimple/Terminator'

Figure 3: Resulting PES for *driver_MODE*=1, *detect_O_E*=1, and *window_system*=1.

Acknowledgement

Main preliminary work is done by our student Paul Buschow who analyzed the power window example and remodeled it with the SES/MB Toolbox for MATLAB for his bachelor thesis.

References

[1] Capilla R, Bosch J. *Binding Time and Evolution*. In Systems and Software Variability Management, edited by R. Capilla, J. Bosch, and K. C. Kang, pp. 57-73. Springer 2013, Berlin Heidelberg, Germany.

[2] Zeigler BP. *Multifaceted Modelling and Discrete Event Simulation*. Cambridge 1984, Academic Press.

[3] Zeigler BP, Kim TG, Praehofer H. *Theory of Modeling and Simulation*. 2nd ed. San Diego 2000, CA, USA, Academic Press.

[4] Schmidt A. *Variant Management in Modeling and Simulation Using the SES/MB Framework*. Ph.D. thesis, Rostock University, Germany. Submitted 10/2018.

[5] The MathWorks Website *Power Window Control Project*. https://de.mathworks.com/help/simulink/examples/power-window-control-project.html, accessed 2018-11-01.

[6] Pawletta T, Schmidt A, Durak U, Zeigler BP. *A Framework for the Metamodeling of Multivariant Systems and Reactive Simulation Model Generation and Execution*. SNE - Simulation Notes Europe. 2018; SNE 28(1): 11-18. doi: 10.11128/sne.28.tn.10402.

[7] SESToPy at GitHub. https://github.com/hendrikfolkerts/sestopy, accessed 2019-03-07.

[8] Folkerts H, Pawletta T, Deatcu C. *A Python Framework for Model Specification and Automatic Model Generation for Multiple Simulators*. In: Proc. of ASIM-Workshop STS/GMMS, Braunschweig, Germany, 21./22., February, 2019, ARGESIM Report 57 & ASIM Mitteilung AM 170, ARGESIM Pub. Vienna/Austria 2019, pp. 69 - 75. (ISBN 978-3-901608-06-3)