Variability Modeling for Engineering Applications

Hendrik Folkerts[1*], Thorsten Pawletta[1], Christina Deatcu[1], Umut Durak[2]

[1]Research Group Computational Engineering and Automation, University of Applied Sciences Wismar
Philipp-Müller-Straße 14, 23966 Wismar, Germany; **hendrik.folkerts@cea-wismar.de*

[2]German Aerospace Center, Lilienthalplatz 7, 38108 Braunschweig, Germany

SNE 27(4), 2017, 167 - 176, DOI: 10.11128/sne.27.tn.10391
Received: December 10, 2017 (Selected ASIM GMMS/STS 2018 Conf. Publ.), Accepted: December 20, 2017
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The System Entity Structure (SES) is a high level approach for variability modeling, particularly in simulation engineering. The SES is under continuous development. In this context, an enhanced framework is introduced that supports dynamic variability evolution using the SES approach and connects the SES to a model base (MB). Using this framework allows building executable models. However, the main focus of this paper is to show how to use the SES to model complex engineering system configurations using a test bench for valves. Specifying the SES, it is clarified, that the system configuration can be decomposed into generalized design patterns. The design patterns were identified during the development of our MATLAB-based SES toolbox for construction and pruning of SES trees and were employed for testing and verification of the respective functionality.

Introduction

Variability modeling is an approach to describe more than one system configuration derived from one underlying configuration. The basic system configuration is often refered to as product line [1]. According to [2] the most popular technique of modeling commonality and variability of products in software engineering is feature modeling. Feature models were introduced by Kang in [3], were subsequently extended, and used in various ways.

An important further development of variability modeling has been the notion of variability in time, known as binding time in product line engineering [4]. Hence, variability can now be realized from design time to runtime.

An approach to variability modeling in simulation engineering is the System Entity Structure (SES) introduced by Zeigler in [5]. The objective was to describe a set of system configurations for a family of systems. In the early nineties the idea of combining the SES with a model base (MB) in order to generate an executable model led to the SES/MB approach [6].

An SES is represented by a tree structure, which describes a set of modular, hierarchical system structures, defines references to basic models in a MB and specifies various parameter settings for the referenced basic models. The classical SES/MB framework only allows static modeling.

To allow dynamic variability modeling and to keep a lean SES tree, the SES/MB framework was extended by several features, such as an interface to the SES and a procedural knowledge representation [7]. In the Research Group Computational Engineering and Automation (RG CEA) at the University of Applied Sciences Wismar a prototype of a SES/MB-based modeling and simulation infrastructure has been developed and implemented within MATLAB/Simulink.

In this paper a short introduction into the extended SES/MB framework is given. It is demonstrated, how the framework can be used to model and simulate a configurable test bench for valves. The main goal is to show that the whole SES can be constructed using elementary design patterns. These in turn can be classified into the categories of the feature relations known from feature models.

1 Background

According to [8], the SES is an ontology, a language with syntax and semantics to represent declarative knowledge. It is particularly suitable for describing system configurations for different application domains.

An SES is represented by a directed tree structure. Objects are represented by nodes which are connected by edges. There are four node types with different properties describing the objects and their relations. Furthermore, there are axioms for defining the SES correctly. Since an SES describes a number of system configurations, the SES tree needs to be pruned to get one particular configuration, which is called Pruned Entity Structure (PES).

The classic SES theory was extended by several researchers over the last decades. Some of these extensions, which are introduced in [7, 9], are used in this paper.

1.1 Node Types

Among the four node types, there are two groups, the entity nodes and the descriptive nodes. Entity nodes describe objects of the real or the imaginary world. The root and the leaves of an SES tree are always entity nodes. Relations between the entity nodes are specified by descriptive nodes.

Descriptive nodes are the genus for aspect nodes, specialization nodes and multi-aspect nodes. Aspect nodes describe how entity nodes can be decomposed into partial entities whereas the taxonomy of an entity is described by specialization nodes. Multi-aspect nodes are a special case of an aspect node with all children being of the same kind.

Each node or edge can have attached variables, also called attributes. For entity nodes, the variables represent properties of the respective object whereas the variables at descriptive nodes specify relations between their parent node and children nodes or decisions for the pruning process. With the extended procedural knowledge representation as described in [7, 9], values of attached variables can be assigned dynamically.

1.2 Axioms

The semantics of the SES are defined by axioms. The types of the nodes have to follow the axiom alternating mode. Every entity node has to be followed by a descriptive node and vice versa. A strict hierarchy is needed. In every path of the tree, a name of a node may occur only once. If nodes in different paths have the same name, they need to have the same variables and isomorphic partial trees. This is called uniformity. Nodes on the same level of hierarchy, called sibling nodes, have to be valid brothers, meaning that sibling nodes must not have the same name.

The axiom of attached variables implies that a node must not have variables of the same name. The axiom of inheritance implies, that during pruning the parent and the child of a specialization combine their attributes. If parent and child have the same attributes, the parent's attributes are overwritten with the child's attributes and their values.

1.3 Extended SES/MB Infrastructure

The SES describing a set of system designs has been associated with the idea of model generation of modular, hierarchical systems from the very beginning (Zeigler, Praehofer, and Kim 2000) which led to the SES/MB approach. Each system design is defined by its system structure and parameter configuration in the SES. The core assets of all system variants are specified as a set of configurable basic models, which are organized in a Model Base (MB).

The classic SES/MB framework defines a set of transformation methods for generating executable simulation models, but automated model generation is not provided. To allow automated generation and execution of models, the SES/MB approach has been extended [7, 9, 10]. These extensions make the SES/MB approach more pragmatic for implementation and to be used in a simulation infrastructure.

Figure 1 depicts the extended SES/MB infrastructure consisting of the SES/MB framework, an Execution Unit, and an Experiment Control. Although the SES/MB approach and its extensions are usually considered in connection with the generation of simulation models, they are generally applicable to modular-hierarchical structured software systems.

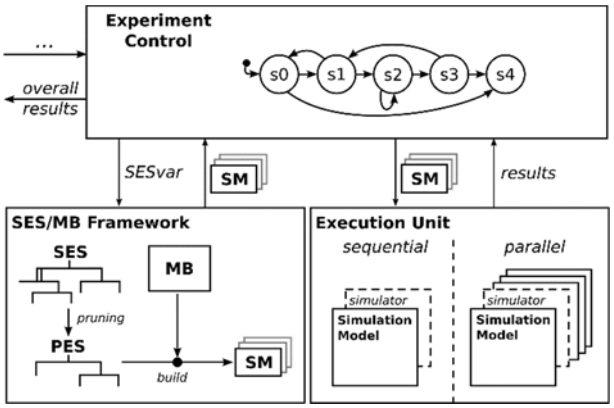

Figure 1: Extended SES/MB-based infrastructure.

Operations.
On the SES, a merge operation is defined allowing two or more SES to be combined. This allows the quick reuse of a once defined SES. The essential operation on the SES is the pruning method. To extract one particular system structure and configuration, the SES needs to be trimmed to a PES. During the pruning process, decisions have to be taken at descriptive nodes. Therefore, rules need to be defined at aspect, multi-aspect and specialization nodes. The specialization rule (specrule) associated with a specialization node determines which child entity shall be selected. Aspect rules (aspectrule) associated with aspect or multi-aspect nodes on the same hierarchy level determine which of the siblings is to be chosen.

Furthermore, cross-tree relations can be expressed by selection constraints. Selection constraints can be used to select a certain entity based on decisions taken anywhere else in the SES tree. Next to the pruning method, another transformation method is the build method. With the help of the build method, an executable model can be built from a PES and basic models organized in an MB. The basic models are specific for a certain simulation software. Therefore, the build method needs to match the simulator used.

Execution Unit and Experiment Control.
For automated and reactive processing of SES models, an execution unit and an overall experiment control unit are added to the framework, as depicted in Figure 1. For automatic generation of different PES, leading to different simulation models, an interface to the SES is needed. This interface can be established using global variables of the SES, called SES Variables (SESvar), which can affect the decisions taken in descriptive nodes during pruning. Thus, a particular system configuration derived from an SES depends on the current settings of the SES variables. The value range of SES variables can be limited by defining semantic conditions, which are checked before pruning to exclude certain system configurations.

By assigning values to the SES variables, the experiment control determines the order and system configurations of executable simulation models (SM) to generate from the SES with the pruning and build operations. Thereby, different variants of the executable simulation models are generated. The experiment control then transmits the SM to the execution unit.

The execution unit links the generated simulation model to the simulator, executes a simulation run and, finally, sends the results back to the experiment control. The results, in turn, can influence the decision of experiment control on how to assign the SES variables next.

Special Attributes.
Combining basic models from the MB leads to the creation of coupled models. In order to describe the structure of the executable model, some nodes need to define couplings. Couplings are properties of descriptive nodes of the type aspect and multi-aspect and consist of pairs of entity names and port names. They describe causal or acausal relationships. Furthermore, for a multi-aspect node, a special variable, numRep, has to be defined representing the number of children to generate when pruning this node. To specify the basic model from the MB an entity node refers to, the mb-attribute is introduced. This special attribute is permitted just for leaf nodes. Finally, for some cases, it is necessary to define priorities among descriptive nodes on the same level of hierarchy in the priority attribute. All values of attributes can be defined by constants or set via SES variables or SES Functions.

SES Functions. The concept of SES Functions (SESfcn) has been introduced to specify complex variability within node attributes with minimal effort and to keep a lean SES tree. Typical examples include the definition of varying coupling relations, varying port numbers of systems or the definition of variable parameter configurations in attributes. During pruning, SES functions are evaluated, often with SES variables as input parameters. For effective coding of SES functions, the implicit attributes parent and children are introduced for each SES node. They encode the parent and children node names, respectively.

1.4 Software Tools for the Extended SES/MB Infrastructure

In the Research Group CEA, a prototype tool for the SES/MB infrastructure was developed, The SES Toolbox for Matlab/Simulink [10]. Currently, SES trees can be defined via a graphical user interface and a concrete variant can be extracted by pruning. The toolbox supports the modeler with plausibility tests during SES construction, graphical representation of the SES, automatic generation of HTML documentation, and other features.

The pruning process can be started from the graphical user interface and, in addition, is implemented to function automatically. Automatic pruning is necessary when using an SES constructed with the toolbox together with the experiment control. Furthermore, there is a prototype Matlab function implementing a build method for the simulation software Simulink, including Sim Events and Simscape, the MatlabDEVS toolbox [11], and for Modelica models. The SES is linked to the appropriate MB with the special mb-attribute of the leaf entity nodes.

Another software tool based on Python3/PyQt5 is under development. The aim is to be more independent from a computing environment and to support a greater number of simulators for building executable simulation models.

2 An Engineering Application Specified with the Extended SES/MB Framework

For development and testing of the pruning algorithm used in the software tools, design patterns were found. Design patterns are rather small SES fragments with a certain behavior when pruned. They can be used to compose complex SES. The design patterns identified can be classified according to the feature relations in feature models. In the context of feature modeling, four kinds of features are used: (i) mandatory features (logical AND), (ii) alternative features (logical XOR), (iii) optional features and, (iv) OR-features (logical OR). In an SES mandatory and alternative features can be expressed in several ways [12].

In order to be able to describe optional features an extension of the SES is used, the NONE element [8]. A NONE element as a leaf entity node means that, if the NONE branch is selected, the entity is not included at all. OR features can only be expressed by the combination of an aspect node with two or more specialization nodes. Each of the specializations needs to have one NONE element as child.

In this section, an engineering example describing a test bench for valves is given, here called industrial plant (*industrialPlant*) for generalization. The SES of the entire plant structure and the corresponding process controls is depicted in Figure 2, but without a detailed specification of coupling attributes.

Since different valve types shall be tested, there is a need for a variability in structure for different test procedures. The example shall demonstrate how a complex SES tree can be composed of design patterns, how the SES can be pruned to extract one variant in the form of a PES, and how an executable model can be built from the PES. The feature relations, except for the OR relation, can be found in the SES. Due to complexity, the SES is split up into several SES, depicted in several figures, to increase clarity and to demonstrate modular SES-based modeling.

2.1 Main SES

The plant is a composition of the plant structure and the necessary process control strategies. Firstly, the plant structure (*plantStructure*) shall be explained. The plant consists of the plant parts for gas (*gas*), liquid (*liquid*), the electrical power supply (*electricalEnergy*) and finally, there are *controllers* to drive the plant. The *gas* and *liquid* entities are decomposed further whereas the electrical energy (*electricalEnergy*) is a merge point to the SES in Figure 6. Merge points are written in bold letters in Figure 2.

As described, merge points allow to combine two SES, which are specified seperately. A merge point is characterized by the same entity node name in both SES, which shall be combined. This implies, that both SES are defined completely and fulfill all axioms.

The mandatory relations described by the aspect node *plantStructureDec* represent the simplest design pattern for pruning, since the resulting PES is identical to the SES. We call it design **pattern #1**. The design patterns are numbered according to the classification found in [12] and pruning is explained for them in the following. One resulting PES representing one test bench configuration is discussed in section 2.8.

Pressure vessels (*pressureVessels*) as well as *compressors*, *valves*, *pumps* and *controllers* are entities to subsume a varying number of identical children. They are the parents of a multiple aspect, so during pruning a number of their respective children is created. The entity *controllers* for example consists of a number of the entity *controller*. The number is defined in the variable numRep. This is another mandatory feature with regard to feature models. It is referred to as design **pattern #2**. Pressure vessel (*pressureVessel*), *compressor*, *valve*, and *pump* are merge points to SES specified later.

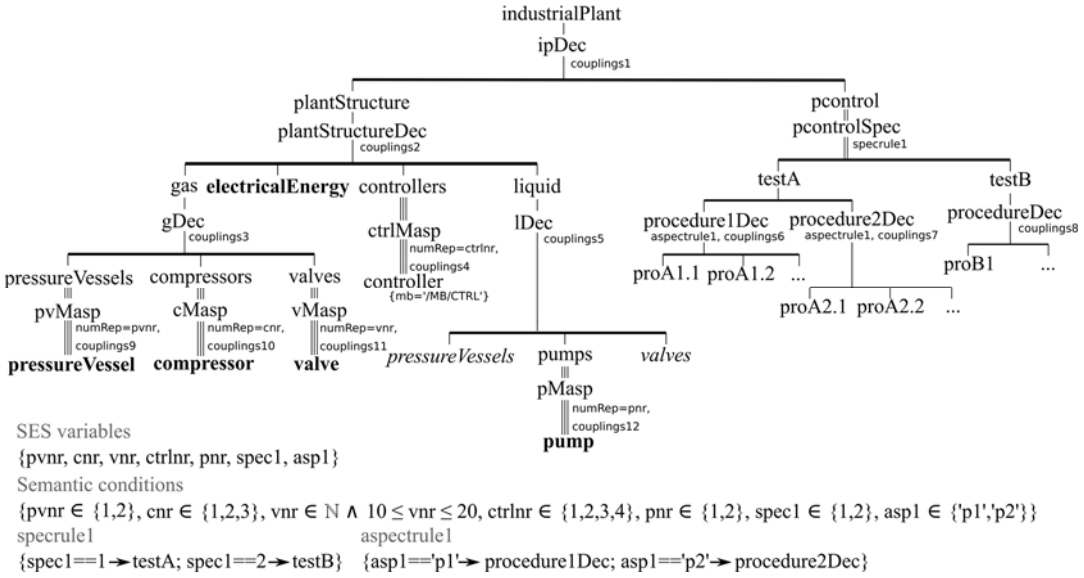

Figure 2: Main SES of the test bench for valves (without detailed coupling definitions).

As described before, one SES axiom urges nodes in different paths with the same name to have the same variables and isomorphic partial trees. Therefore the entities *pressureVessels* and *valves* in the branch for the liquid (written italic in Figure 2) do not need to be specified since they are specified for the partial tree defining the gas.

Before listing the sub SES for merge points, the process control part in the main SES shall be explained. In a derived variant of the SES, either *testA* or *testB* can be part of the PES to specify a particular system configuration. This is an alternative selection, called design **pattern #4**. Therefore, the entities *testA* and *testB* are specializations of the process control that is represented by the entity *pcontrol*. For decision, which test to execute, a specialization rule (specrule) has to be defined at the specialization node *pcontrolSpec*. During pruning the specrule is interpreted and the selected *testA* or *testB* entity is united with the parent entity *pcontrol*.

If *testA* is selected, there are two possible main procedures which can be taken alternatively. The decision is taken by evaluation of aspectrule1 at *procedure1Dec* and *procedure2Dec*. This design pattern for an alternative selection is referred to as **pattern #5**. The decision taken leads to a number of elementary procedures (e.g. *proA1.1*...*proA1.n*) specific for the selected superordinate main procedure. For testB there is only one main procedure leading to a number of elementary procedures as explained in design **pattern #1**.

The SES in Figure 2 demonstrates the use of SES variables (SESvars) for dynamical generation of different system configurations. SESvars can contain any type of value. At nodes needing a decision during pruning, the corresponding attribute or rule can be defined based on an SESvar, so each value can be assigned dynamically. The value range of every SESvar should be restricted using semantic conditions. A semantic condition needs to evaluate to a logical value and can contain more than one logical expression. SESvars are assigned to the numRep attributes at the multi-aspect nodes. There are the SESvars *spec* and *asp* used in the specrule and the aspectrule respectively. If the SESvar *spec* equals 1, *testA* is taken, if *spec* equals 2, *testB* is selected during pruning. The SESvar *asp* can take one of two possible strings as value. The aspectrule defines that, if *asp* equals the string p1, *procedure1Dec* is selected, and if *asp* equals the string p2, *procedure2Dec* is taken.

The aspect and multi-aspect nodes need the special attribute couplings as seen in Figure 2. They describe the relations of entity nodes. An example for a coupling definition is given in section 2.9. In this context the usage of SES functions is illustrated as well.

2.2 Sub SES: Pressure Vessel

In Figure 3 the pressure vessel (*pressureVessel*) is specified. Every pressure vessel has the two attributes volume and maxPressure. Creating the PES depending on the specrule defined, either *steel* or *aluminum* is taken as material, which is united with the *pressureVessel* entity.

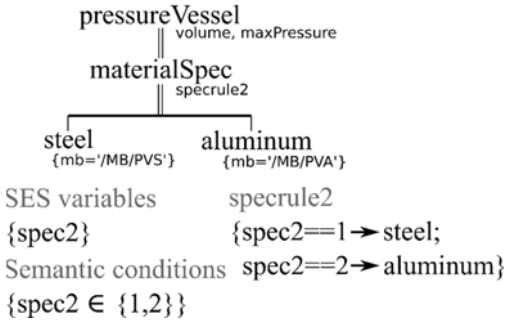

Figure 3: SES of a pressure vessel.

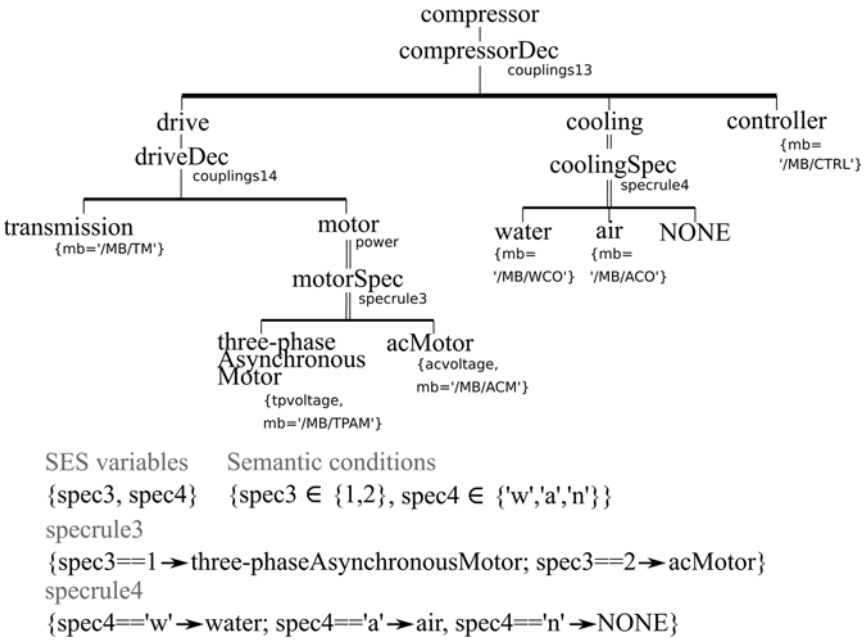

Figure 4: SES of a compressor.

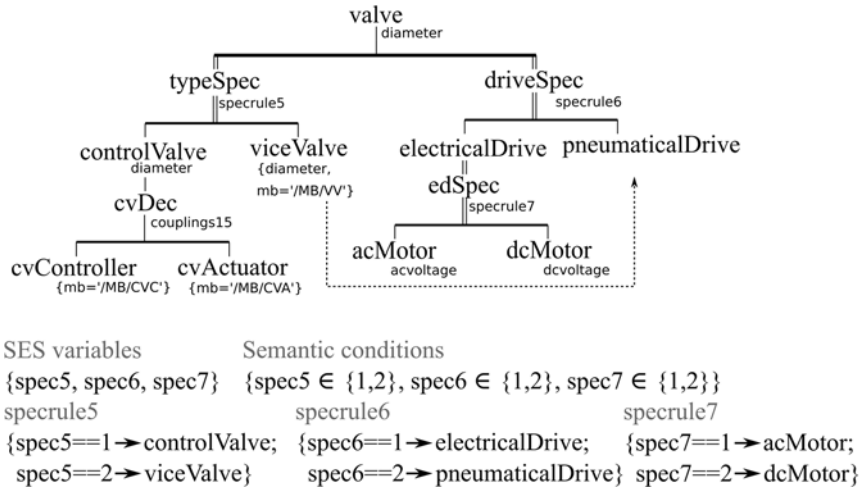

Figure 5: SES of a valve.

The mb-attribute of the leaf entities refers to a basic model in the MB, which is depicted in Figure 8. During pruning attributes are inherited to the parent, which is renamed and has three attributes afterwards. This is another example for the design **pattern #4** with the addition of attribute inheritance. As in the main SES, the decision at the specialization is taken based on the specrule and the current values of SES variables.

2.3 Sub SES: Compressor

The details of the compressor can be seen in Figure 4. A *compressor* is composed of the *drive*, the *cooling* and a *controller*. The *drive* is composed of the *motor* and a *transmission*. This is an SES construction as described in design **pattern #1**. The motor can be specified as a three-phase asynchronous motor or an ac motor, which refer to a model in the MB with their mb-attribute. During pruning, this part is resolved as described for the pressure vessel in design **pattern #4**.

The cooling is an optional element. Since it is a specialization with a NONE element, the pruning process resolves the cooling either to water cooling, air cooling or no cooling at all. The optional tree section is referred to as design **pattern #8**.

2.4 Sub SES: Valve

Valves are needed for the plant parts for gases as well as for liquids and are only defined in the partial SES defining the gas as described before. A valve is of a type and needs to have a drive. The specialization siblings *type Spec* and *driveSpec* in Figure 5 can be classified as mandatory feature and are referred to as design **pattern #3**.

The entity *electricalDrive* is specialized into *acMotor* and *dcMotor* according to design **pattern #4**. During the pruning process all specializations are resolved, attributes are inherited and the entity *valve* is renamed with the selected type, drive and, if the *electricalDrive* is selected, the motor type. Two specialization nodes in one path as seen in this example is a combined design pattern referred to as design **pattern #10** and is showing attribute inheritance among several layers. In section 2.8 the inheritance of names and attributes is explained in detail. The *controlValve* is composed of the *cvController* and the *cvActuator*. Seen from the top entity *valve*, this is another combined design pattern consisting of a specialization with a succeeding aspect. We call it design **pattern #11**. It shows that during pruning the specialization *typeSpec* is resolved according to design **pattern #4** and the aspect *cvDec* is not changed like in design **pattern #1**. The dotted line in Figure 5 is a selection constraint for cross-tree relations, which forces the use of a *pneumaticalDrive*, if a *viceValve* is selected. It may be noticed, that in Figure 5 not all leaf nodes have an mb-attribute. Either the leaf nodes in the branch of the *typeSpec* entity or the leaf nodes in the branch of the *driveSpec* are allowed to have mb-attributes. The other branch can only configure the basic model. In this example the mb-attribute is set in the *typeSpec* branch, while the configuration is set in the *driveSpec* branch. During pruning identical attributes are overwritten in parent nodes, so would be the mb-attribute if defined for both branches.

2.5 Sub SES: Electrical Energy

The SES describing the electrical energy is shown in Figure 6. The electrical energy (*electricalEnergy*) can be generated with multiple generators specified in the variable numRep or the electrical grid (*gridDec*) is used.

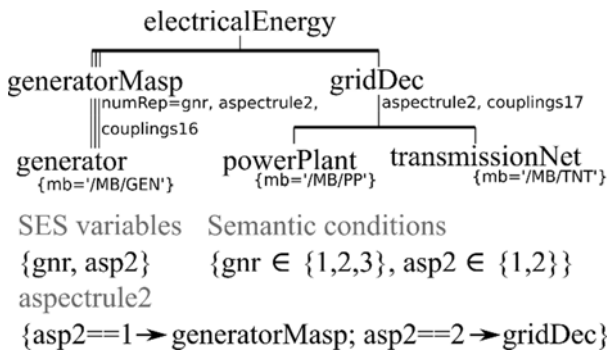

Figure 6: SES of the electrical energy

The grid is decomposed into a power plant (*powerPlant*) and the transmission net (*transmissionNet*). The decision, whether the generator or the grid is selected, is taken in aspectrule2. This alternative structure created of siblings of aspect and multi-aspect nodes is called design **pattern #7**.

2.6 Sub SES: Pump

The *pump* can be built with the SES depicted in Figure 7. It is decomposed into the *motor*, the *wheel* and a *controller* just like design **pattern #1**. The motor does not need to be specified since it is already defined in the SES of the compressor (see Figure 4).

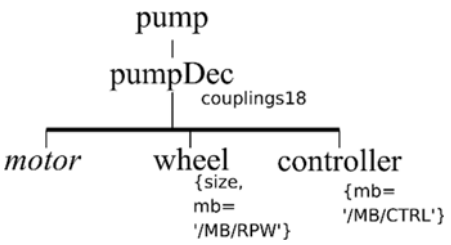

Figure 7: SES of a pump.

2.7 Model Base

For model generation a library containing the configurable basic models specified in the mb-attribute of leaf nodes is needed. A possible model base (MB) for this example is given in Figure 8. The MB contains the dynamic models for simulation of the respective devices. Every model needs input and output ports for building coupling relations. A basic model in the MB can be composed of several models as depicted for the *TNT* model.

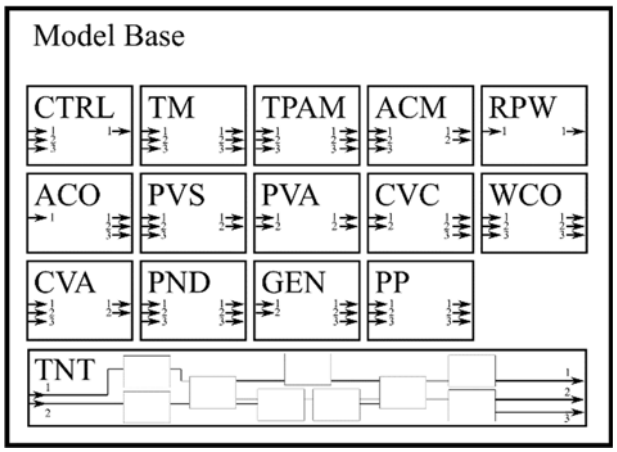

Figure 8: Model Base for this example.

2.8 Deriving a PES

Since an SES specifies a number of different system configurations, the SES has to be transformed into a PES by pruning in order to get one particular system configuration. Figure 9 shows a valid SESvars setting and the derived PES, which represents one variant of the configurable test bench for valves. Before pruning, values are assigned to the SESvars and checked for validity using the semantic conditions. The process of resolving all necessary decisions is described in the following.

First, it shall be described how to derive a particular variant from the main SES in Figure 2. Since the resulting PES for pruning **pattern #1** is identical to the SES, the composition of the plant (*industrialPlant*) as well as the composition of the plant structure (*plantStructure*), the gas (*gas*) and the liquid (*liquid*) are part of the PES. The multi-aspect nodes *pvMasp*, *cMasp*, *vMasp*, *ctrlMasp*, and *pMasp* are resolved following design **pattern #2** by converting them into aspect nodes with the name extension *Dec* and a certain number of children is created as defined in numRep. The number of each child is appended to the nodename of the child.

For the electrical energy (*electricalEnergy*) the selection for *generatorMasp* is taken in the aspectrule2 (design **pattern #7**) and the respective children (*generator*) are created as in design **pattern #2**.

The pressure vessel is pruned according to design **pattern #4**. As material of the pressure vessel (*pressureVessel*) aluminum is selected by evaluating specrule2. The name of the selected entity (*aluminum*) is united with the parent of the specialization (*pressureVessel*) to *aluminum_pressureVessel*, attributes of the child are inherited to the renamed parent and the specialization node *materialSpec* is deleted.

The compressor drive (*drive*) can be pruned using design **pattern #1** and design **pattern #4** as before. The cooling is a specialization evaluating to *NONE* according to design **pattern #8**. The NONE node is depicted in Figure 9 for understanding, but has no relevance for the succeeding model generation.

For generating the PES of a valve based on the SES in Figure 5, both specialization siblings have to be resolved, the resulting names are connected and the attributes are inherited (design **pattern #3**). It depends on the pruning algorithm used, which of both specializations is pruned first. There is no explicit rule. In this example the pruning is started with the specialization *typeSpec*. The type *controlValve* is selected and an intermediate parent entity *controlValve_valve1* can be assumed. The value of the attribute diameter in the parent node is replaced with the value of the same attribute in the child according to design **pattern #4**.

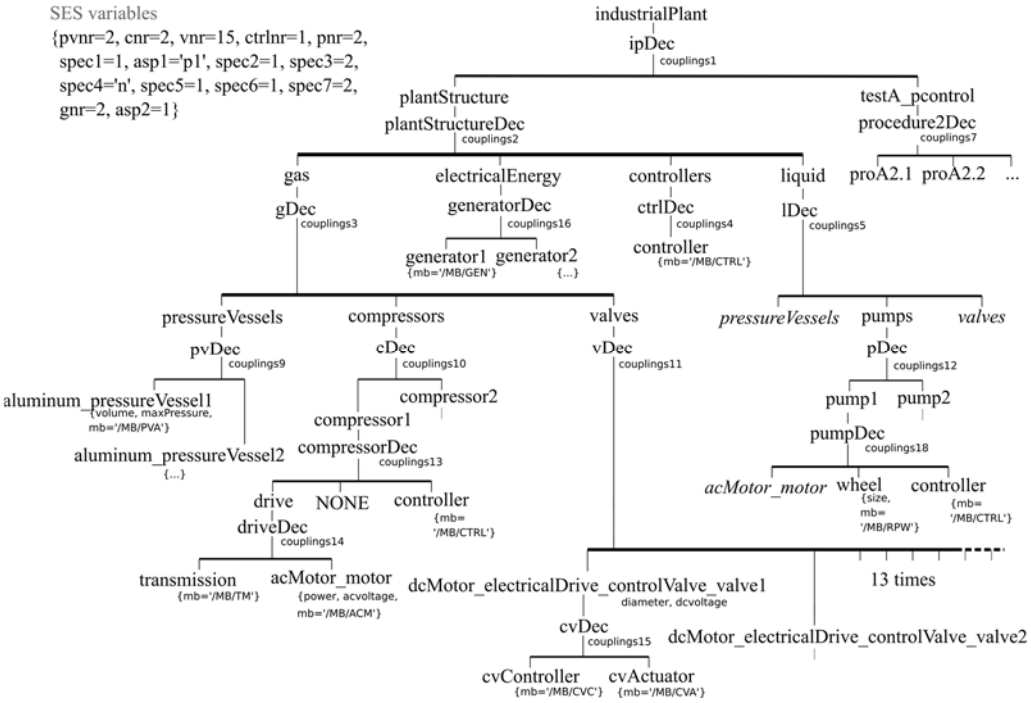

Figure 9: PES describing one particular system configuration derived based on the current SESvars settings.

The tree beginning with *cvDec* is a composition, which is not changed during pruning (design **pattern #1**).

Since *electricalDrive* is selected in *driveSpec*, both of the specializations in one path (design **pattern #10**) *driveSpec* and *edSpec* need to be resolved like in design **pattern #4**. The entity node *dcMotor* is selected, the attribute dcvoltage is moved to the top entity, which is renamed to *dcMotor_electricalDrive_controlValve_valve1* according to the selections.

The entities *pressureVessels* and *valves* in the branch for the liquid (written italic in Figure 9) are specified for the partial tree defining the gas and do not need to be specified again. That implies, that in this design the pressure vessels and the valves have to be of the same type and number and need to have the same properties for both plant parts.

A *pump* as depicted in Figure 7 is a composition of a *motor*, a *wheel* and a *controller* and is pruned according to design **pattern #1**. However, since the motor of the pump is the same entity as the motor of the compressor caused by the same name in the SES, in the PES both nodes need to have the same name as well.

The process control (*pcontrol*) is derived by evaluation of specrule1 like in design **pattern #4** and the aspectrule1 for aspect siblings (design **pattern #5**). The entity node *testA* is selected and united with *pcontrol*. Finally the aspect node *procedure2Dec* is selected, which is the child of the renamed *testA_pcontrol*.

Summarized, in this configuration the industrial plant has four pressure vessels, two compressors, thirty valves, two pumps, two generators and one controller for the plant. Each device of these device groups has the same structure and is of the same type.

However, in the case of the motors of the compressors and pumps it makes sense to relax the axiom of uniformity, so that different values can be assigned to the power attribute. The relevance to relax the uniformity attribute is discussed in [7] comprehensively.

Since the PES is a decision-free tree structure, a particular executable model can be generated based on its information and using basic models from the MB.

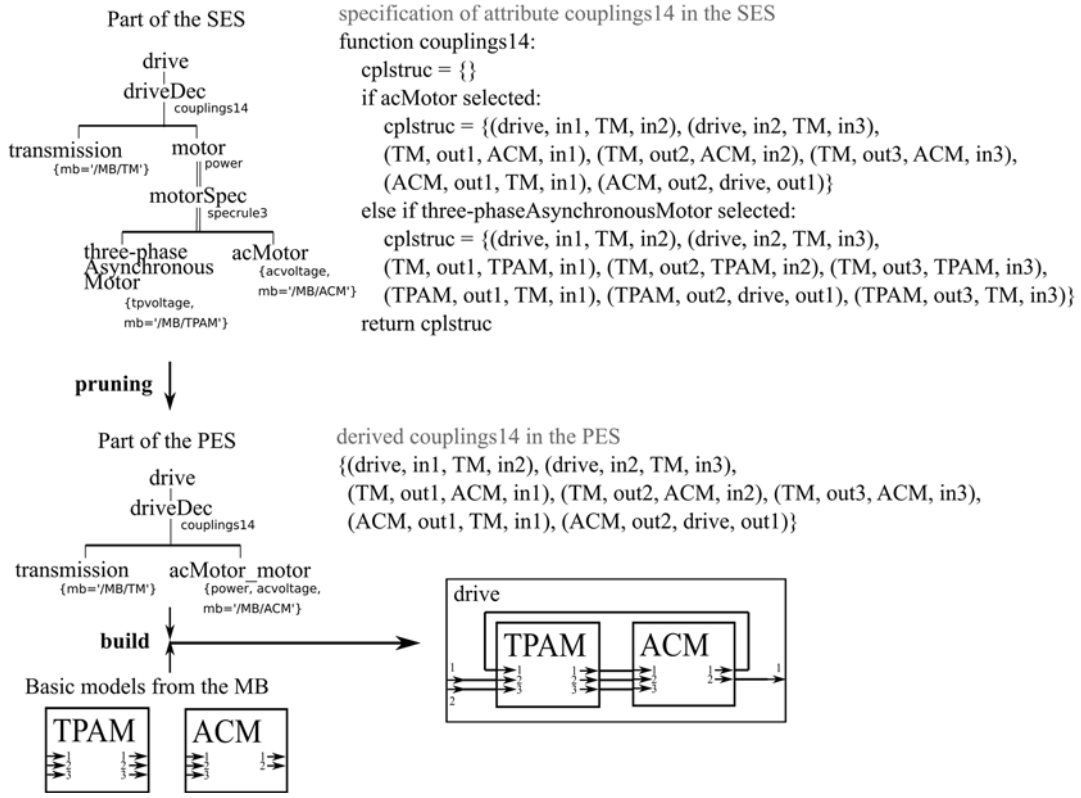

Figure 10: Specification of dynamic couplings in an SES, derived static couplings in the PES, and resulting executable model.

2.9 Building the Executable Model

For the generation of the executable model, the coupling attributes are especially important, as discussed in section 1.3. Examples on how couplings are specified and resolved are depicted in Figure 10 according to the SES part of a drive of a compressor in Figure 4.

In the SES couplings often have to be defined dynamically, since the SES describes several configuration variants. During pruning, one particular configuration with static couplings is derived, so that an executable model can be built. Dynamic specification can be done using SESfcns as illustrated in pseudocode in Figure 10. In the example, the attribute couplings14 in the SES has to specify the coupling relations for both types of basic motor models with a different number of ports. In the PES the concrete static couplings for the selected basic models are derived, in this case for the basic model *ACM*.

Particularly, when defining the couplings for a multi-aspect node or for nodes referring to basic models, whose port numbers vary depending on the selected configuration, defining the couplings via SESfcns is best practice.

3 Conclusion

This paper discusses how to use the extended SES/MB framework in order to describe and build a set of simulation models of a complex configurable engineering application by the example of a test bench for valves. It is shown, that previously found design patterns are appropriate to specify a complex engineering problem. Such engineering problems can be modeled using the proposed software tools for the extended SES/MB infrastructure, executable models for different simulators can be built and finally simulated.

Acknowledgements

The authors acknowledge the grant from the German Science Foundation, DFG (PA 631/2). Moreover, the authors would like to thank Peter Junglas, who contributed valuable work to the development of a generic model builder for the SES toolbox; Daniel Pascheka, who implemented a first version of the graphical editor within MATLAB; Birger Freymann, who redesigned the editor and implemented a model builder for the MatlabDEVS toolbox and our former colleague, Tobias Schwatinski, who provided preliminary work. Last, but not least, we would like to thank Bernie Zeigler for motivating and supporting our work.

References

[1] Alt O. Variantenmanagement. In: *Modellbasierte Systementwicklung mit SysML*. Munich: Hanser, 2012. p 6 total pages of chapter.

[2] Kang KC, Lee H. Variability Modeling. In: Capilla, R., Bosch, J., Kang, K. C., editors. *Systems and Software Variability Management.* Berlin: Springer; 2013. p 7 total pages of chapter.

[3] Kang KC, Cohen SG, Hess JA, Nowak WE, Peterson AS. Feature-Oriented Domain Analysis (FODA) Feasibility Study. *Technical Report CMU/SEI-90-TR-21, ESD-90-TR-222, SE Inst. Carnegie Mellon Univ. Pittsburgh/PA, USA.* 1990.161 p.

[4] Kang KC, Lee H. Binding Time and Evolution. In: Capilla R, Bosch J, Kang KC. editors. *Systems and Software Variability Management.* Berlin: Springer; 2013. p 7 total pages of chapter.

[5] Zeigler BP. *Multifaceted Modelling and Discrete Event Simulation.* 2nd ed. London: Academic Pr.; 1984. 372 p.

[6] Zeigler BP, Kim TG, Praehofer H. *Theory of Modeling and Simulation*. 2nd ed. Cambridge: Academic Pr.; 2000. 510 p.

[7] Pawletta T, Schmidt A, Zeigler BP, Durak U. Extended Variability Modeling Using System Entity Structure Ontology within MATLAB/Simulink. In Proc. of Spring Simulation Multi-Conference 2016. *Spring Simulation Multi-Conference*; 2016 Apr; Pasadena/CA, USA. SCS Int. p 62-69.

[8] Zeigler A, Hammonds PE. *Modeling and Simulation-Based Data Engineering.* 1st ed. Academic Pr.; 2007. 448 p.

[9] Schmidt A, Durak U, Pawletta T. Model-Based Testing Methodology Using System Entity Structures for MATLAB/Simulink Models. *SIMULATION: Transactions of The Society for Modeling and Simulation International.* 2016; 92(8): 729-746.

[10] RG CEA. *The SES Toolbox for MATLAB/SIMULINK Website.* http://www.cea-wismar.de/tbx/SES_Tbx/.

[11] RG CEA. *The PDEVS Toolbox for MATLAB Website.* http://www.cea-wismar.de/tbx/DEVS_Tbx/MatlabDEVS_Tbx.html.

[12] Deatcu C, Folkerts H, Pawletta T, Durak U. Design Patterns for Variability Modeling Using SES Ontology. In Proc. of Spring Simulation Multi-Conference 2018. *Spring Simulation Multi-Conference*; 2018 Apr; Baltimore/MD, USA. SCS Int. 10 p. (paper submitted).