AUTOMATIC GENERATION OF SIMULATION MODELS USING AUTOMATED, REACTIVE PRUNING OF SYSTEM ENTITY STRUCTURES

Hendrik Folkerts
Thorsten Pawletta
Christina Deatcu

Research Group CEA
University of Applied Sciences Wismar
Wismar, D-23966, Germany
hendrik.folkerts@cea-wismar.de
{thorsten.pawletta,christina.deatcu}@hs-wismar.de

Bernard P. Zeigler

RTSync Corporation
and ACIMS
4425 E Agave Rd. Suite 106 Phoenix, AZ 85044, USA
zeigler@rtsync.com

ABSTRACT

System Entity Structures (SES) are used to define families of systems. In this context they are employed in combination with a model base (MB) to describe a set of simulation models. Using a framework, simulation models are generated from the SES/MB in a goal-oriented manner, executed and their results analyzed. The entire process is automated, iterative and reactive. A concrete system variant in the family of systems is derived by pruning an SES. For the process of automated model generation, it is necessary to provide automatic pruning mechanisms. Especially for some types of nodes comprised in an SES, automatic pruning is a big challenge. Here, the challenges that arise when pruning SES containing hierarchies of multi-aspect and specialization nodes are analyzed and a solution is proposed.

Keywords: System Entity Structures, automated pruning, restructuring of SES.

1 INTRODUCTION

Variability modeling can be seen as an approach to describe more than one system configuration. A system configuration is characterized by a specific system structure and parameterization of its components. According to Capilla, Bosch, and Kang (2013), 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 known from the eighties. One of the first high level approaches for variability modeling was introduced with the System Entity Structure (SES) by Zeigler (1984) and is continuously further developed to this day (Rozenblit and Huang 1991; Zeigler and Hammonds 2007; Zeigler and Sarjoughian 2013; Santucci, Capocchi, and Zeigler 2016; Schmidt 2019). The SES describes a set of system configurations. It was early connected with a model base (MB) organizing basic models which describe dynamic behavior (Rozenblit and Zeigler 1993; Zeigler, Kim, and Praehofer 2000). With the SES/MB approach a set of simulation models (SM) for a family of systems can be generated.

An SES is represented by a tree structure comprising entity nodes, descriptive nodes and attributes. Entity nodes describe an object of the real or imaginary world. Descriptive nodes describe the relations among at least two entities and are divided into aspect, multi-aspect and specialization nodes. An aspect node represents the composition of an entity. The multi-aspect node is a special aspect node and describes the decomposition of an entity into several entities of the same type. The specialization node describes the taxonomy of an entity. Descriptive nodes can specify variation points in context of variability modeling using specific rules.

In engineering, simulation of a system to find a best design is a common approach (Maria 1997). In many engineering problems a huge number of different design variants need to be examined, e.g. in the automotive sector (Oster 2012; Grönniger et al. 2014). Models in this field usually follow a modular-hierarchical structure. Due to the high number of variants the modeling as well as the execution of simulation experiments is a time consuming, error prone, and thus expensive process. Considering e.g. a control development there can be a fairly high number of different system structures and controller configurations to achieve an optimal design. The variety of different system structures and parameter settings can be coded in an SES and the dynamic behavior can be implemented in basic models that are organized in an MB. However, with the increasing complexity and variety of technical systems, the need for an automated search for the best system design increases.

The derivation of one single system configuration coded in an SES is called *pruning*. The resulting design describing a single system configuration comprising structure as well as parameters settings is called Pruned Entity Structure (PES). During pruning all variation points in the SES need to be resolved. The pruning can be done interactively or automated. When doing the pruning interactively the user needs to have the knowledge on how to find decisions at the variation points. Pruning an SES interactively means to manually decide at each variation point to find a certain system configuration. This is a time intensive and error-prone process which is hard to carry out (Zeigler and Sarjoughian 2013). Thus, the automation of the pruning process is desirable. One approach for automated pruning is the enumerative pruning as described in Zeigler and Sarjoughian (2013). When following the approach of enumerative pruning all PES coded in the SES are produced once. Due to the high number of variants in engineering problems the interactive pruning is often not feasible, while enumerative pruning can be neglected since this approach is not goal-directed. A goal-directed automation of the pruning and model building process requires an extension of the SES/MB approach. Knowledge on how to find decisions to resolve the variation points during pruning needs to be defined before starting the pruning process. In Zeigler and Sarjoughian (2013) the suggestion is made to specify rules for pruning in a script. Another approach is the extension of the SES with features to specify all knowledge necessary for automatic pruning in the SES as suggested in Pawletta et al. (2016). This paper builds on these extended features. In Deatcu et al. (2018) some basic design patterns on how to construct SES and prune them automatically are evaluated. However, SES composed of hierarchical multi-aspect nodes and multi-aspect nodes in combination with specialization nodes are not studied.

Multi-aspect nodes and their combination with specialization nodes provide a powerful modeling mechanism. However, this can lead to problems regarding an automated, goal-oriented pruning of SES. A systematic study of possible problems is the focus of this paper. Prior to this, Section 2 briefly introduces the Extended SES/MB (eSES/MB) framework to promote the understanding of automated, goal-oriented model generation. Section 3 then studies step-by-step modeling with multi-aspect nodes and hierarchies of multi-aspects in combination with specializations. Problems of such SES structures regarding automated pruning are analyzed and possible solutions are proposed. Finally, in Section 4, reference is made to some other related work, followed by a conclusion.

2 EXTENDED SES/MB FRAMEWORK

In Zeigler, Kim, and Praehofer (2000) a general framework for the SES/MB approach was introduced. In the *modeling phase* a set of system configurations for a family of systems is coded as an SES or in a set of SES, which can be merged according to Zeigler and Hammonds (2007). The system behavior is mapped in basic models, which can be composed in a modular-hierarchical manner. The basic models are organized in an

MB. In addition, the framework defines a pruning method for deriving a single system configuration from an SES. The resulting structure is a PES which contains all information necessary for model generation. Based on the configuration information in the PES and basic models from the MB a simulation model (SM) is generated by a build method.

The extended SES/MB (eSES/MB) framework, as depicted in Figure 1, was introduced in Pawletta et al. (2016) and Schmidt, Durak, and Pawletta (2016) for the automation of goal-directed simulation experiments using different system configurations. That means, experiments with different system structures and parameterizations of components can be defined, carried out, and evaluated automatically.

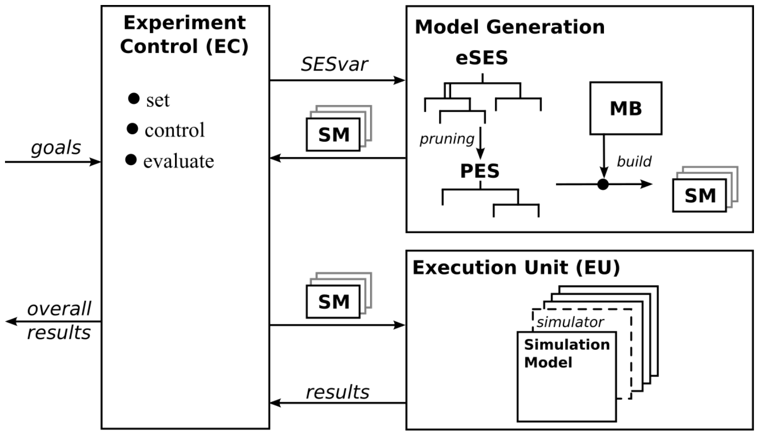


Figure 1: Extended SES/MB framework based on Pawletta et al. (2018).

The main objective of the classic SES/MB framework is an interactive model generation process. Its general structure corresponds with the *Model Generation* component in Figure 1. The definition of experiments and the simulation of generated models are not covered in the classic SES/MB framework and are first introduced with the eSES/MB framework. The extended framework requires extensions of the SES approach (eSES), of which the main ones will be discussed briefly. Although reference is made to eSES below, the term SES will continue to be used for linguistic simplification. The integration of the SES/MB approach into an infrastructure for automated simulation experiments requires an interface to the SES and its methods to derive goal-driven system configurations and to generate SM. The interface to the SES is established by a set of variables with a global scope, the SES variables (SESvar). In addition, global functions can be defined on the SES to allow the definition of procedural knowledge. They are called SES functions (SESfcn). Some types of variability, such as dynamic coupling relations of components, can be described more easily with SESfcns (Folkerts et al. 2019). For automatic pruning, selection rules at descriptive nodes need to be defined, such as *aspectrules* for aspect and multi-aspect siblings or *specrules* at specialization nodes. Like all other node attributes of the SES, the selection rules can use SESvars and SESfcns to specify variable rules. Current values must be assigned to the SES variables before pruning. SES function calls to calculate variable node attributes are executed during pruning. A more detailed explanation of the SES extensions can be found in Deatcu et al. (2018) and Folkerts et al. (2019).

During the *modeling phase* the user specifies a set of system configurations in the SES and creates a simulator specific MB. Moreover, the user has to specify the overall goal of an experiment and the logic for the Experiment Control (EC), which governs the automated, goal-oriented experimentation process. In the *execution phase*, an experiment cycle starts with the assignment of current values to the SESvars by the EC. Then, the *model generation* part derives a certain system configuration as a PES and generates an SM. The EC optionally adds execution parameters to the SM and starts the simulation execution by the EU. The EU controls the simulation execution in a target simulator. The results of the execution process are returned to the EC, where they are evaluated. Based on the result of evaluation, the EC reactively decides whether a further experiment cycle is started or whether the current experiment is terminated. Prototypes of the eSES/MB framework were implemented using MATLAB (Pawletta et al. 2016) and the Python language (Folkerts et al. 2019).

Methods for automatic pruning of SES are a crucial part in the eSES/MB framework and a set of arising problems for complex hierarchies are discussed in this paper.

3 MODELING AND PRUNING OF SES WITH MULTI-ASPECT NODES

Multi-aspect nodes and their combination with specialization nodes provide powerful mechanisms for variability modeling of system configurations. However, structures with multi-aspect nodes and succeeding specialization nodes or with several hierarchical multi-aspect nodes in a path have proved to be hard to be pruned automatically. Multi-aspect nodes define an attribute *number of replications*, named *numRep*, which specifies how many incarnations of the succeeding entity have to be generated. In contrast to the other descriptive nodes, pruning results in a tree expansion. During pruning, at a multi-aspect node parts of the tree are not cut away, but the tree is expanded by the number of generated children. In Zeigler and Hammonds (2007) it was proposed to restructure an SES in order to avoid hierarchies of multi-aspects or their combination with specializations. Restructuring an SES means to find a tree structure that describes the same set of system configurations. The restructured SES might be pruned more easily. However, when restructuring the SES new attributes may be necessary due to the changed structure. Automated restructuring in the context of automated pruning is difficult, because it is not specified how values are to be assigned to the new additional attributes. In the following subsections solutions for an automated pruning of SES trees with hierarchical multi-aspects or combinations of multi-aspects with specializations are discussed. To illustrate this, an artificial example without reference to the system simulation is used and extended step by step. The term *pool* is subsequently used for a related set of resources, such as a set of computers in a laboratory. Consequently, the term *pools* refers to a set of *pool*s.

3.1 Single Multi-Aspect

In Figure 2 the simplest case of an SES with a single multi-aspect node is depicted. The SES specifies a set of systems with a varying number of *pools*. The root entity *pool*s is followed by the multi-aspect node *poolsMASP* with the attribute *numRep = NumPools*, which describes the varying number of pools. In this example, the numRep attribute is set by the SESvar *NumPools*, which defines the input interface of this SES. According to the *Semantic Condition* the SESvar *NumPools* has to be element of $\mathbb{N}$. The multi-aspect node is followed by the entity *pool*. This entity represents a kind of class from which replications are generated during pruning. Due to this property, the entity following a multi-aspect is referred as the generating entity in Zeigler and Hammonds (2007). We propose that in the context of automatic pruning an attribute beginning with an underscore and followed by the name of the entity is added to a generating entity, in this case the attribute is *_pool*. The value of this attribute remains undefined in the SES. The reason for introducing this underscore attribute is pointed out in the next subsections.

Before pruning, values must be assigned to all SESvars of the input interface. In the example in Figure 2, the constant value two is assigned to the SESvar *NumPools*. Pruning a multi-aspect node, a number of entities according to the numRep attribute is created and the multi-aspect is converted to an aspect node. Accordingly, the PES in Figure 2 describes a system configuration *pools* consisting of two subpools, named *pool_1* and *pool_2*. Additionally, the number of each created entity is assigned to its attribute *_pool*.

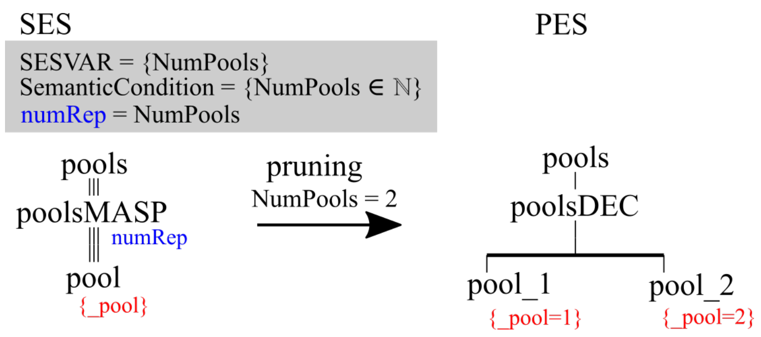


Figure 2: SES with multi-aspect and a possible PES.

3.2 Multi-Aspect with Succeeding Specialization

In this subsection the example from Section 3.1 is expanded by a further layer. With reference to the considered example, an entity *pool* shall be of type *desktop* or type *notebook*. Therefore, as depicted in Figure 3, the SES additionally specifies a specialization node *poolSPEC*, which has the entities *notebooks* and *desktops*. While the numRep attribute at the multi-aspect *poolMASP* is defined by the SESvar *NumPools* as before, the specialization *poolSPEC* defines a *specrule* using an SESfcn named *fun*. The SESfcn defines the selection depending on the additional SESvar *Types* and the attribute *_pool* at the generating entity *pool*. According to the *Semantic Condition* the SESvar *Types* is a vector with elements 'nb' or 'dt' and it needs to have as many elements as the value of the SESvar *NumPools*.

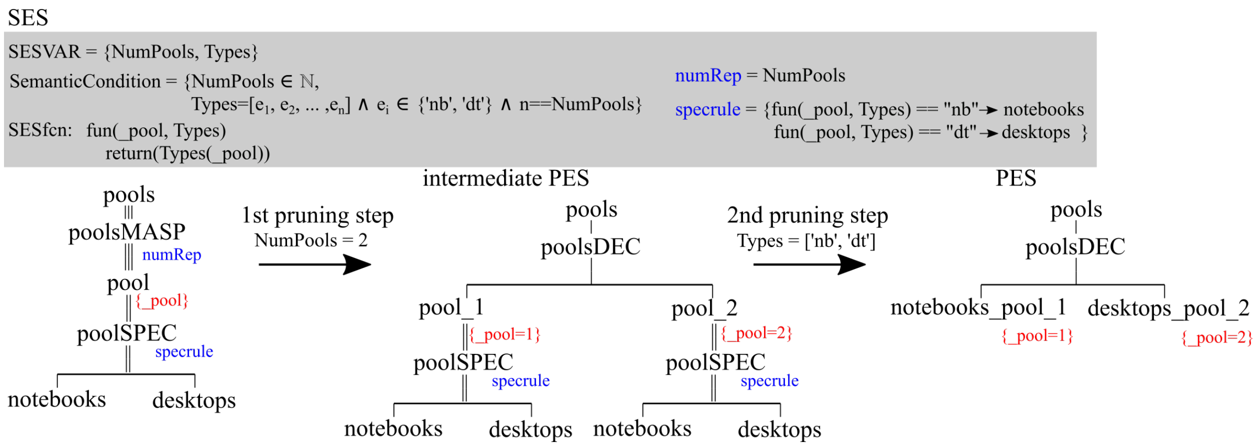
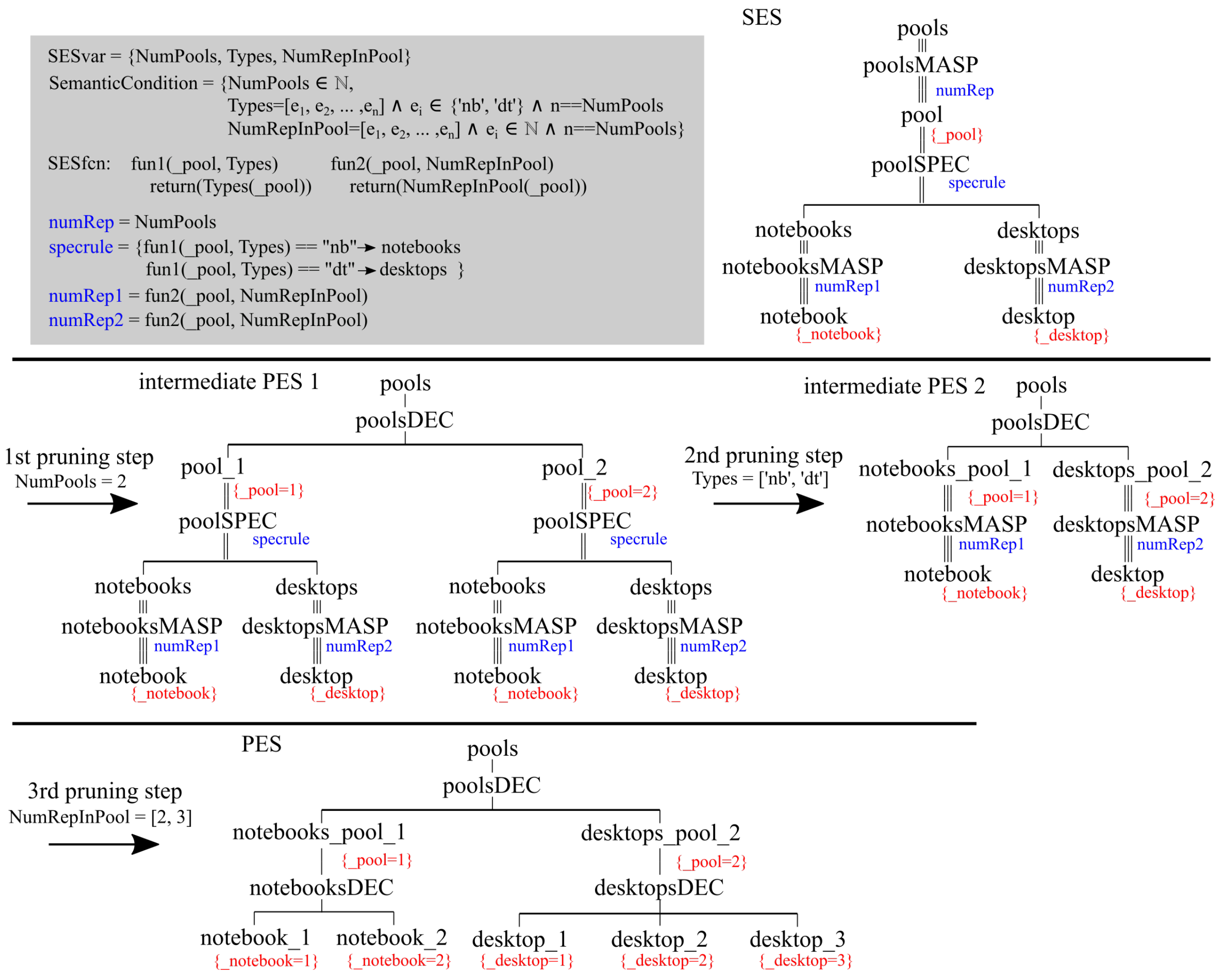


Figure 3: SES with multi-aspect, succeeding specialization, and a possible PES.

In Figure 3 the pruning process with all intermediate steps is depicted. In the first step a number of *pools* is generated analogous to Section 3.1. In the second step the specialization *poolSPEC* is resolved. Specialization nodes express a kind of variation point defining an xor-selection. The selected child in a specialization is united with the parent node of the specialization. Other children as well as the specialization node itself are cut away. In this example depending on the value in the attribute *_pool* the SESfcn evaluates either to 'nb' or to 'dt'. According to its result the *pool* entities specialize either to *notebooks_pool* or to *desktops_pool*.

3.3 Multi-Aspect with Succeeding Specialization and Multi-Aspects

The SES from Section 3.2 is expanded by one more layer. For each specialized *pool* the number of desktops or notebooks can be defined. As depicted in Figure 4, in the SES the *notebooks* and the *desktops* entities are now followed by multi-aspect nodes *notebooksMASP* and *desktopsMASP* respectively. They specify the possible number of the *notebook*s and *desktop*s in each pool. The number of entities is variable. Therefore, the attributes *numRep1* and *numRep2* need to be set dynamically using an SESfcn, named *fun2*. The SESfcn returns the number of elements for a pool. The introduced attribute *_pool* and the SESvar *NumRepInPool* are passed as parameters. *NumRepInPool* is a vector with elements of ℕ and the number of elements has to be equal to the SESvar *NumPools*, as defined in the *Semantic Condition*.



SESvar = {NumPools, Types, NumRepInPool}
SemanticCondition = {NumPools ∈ ℕ,
Types=$[e_1, e_2, ... ,e_n]$ ∧ e_i ∈ {'nb', 'dt'} ∧ n==NumPools
NumRepInPool=$[e_1, e_2, ... ,e_n]$ ∧ e_i ∈ ℕ ∧ n==NumPools}

SESfcn: fun1(_pool, Types) return(Types(_pool)) fun2(_pool, NumRepInPool) return(NumRepInPool(_pool))

numRep = NumPools
specrule = {fun1(_pool, Types) == "nb" → notebooks
fun1(_pool, Types) == "dt" → desktops }
numRep1 = fun2(_pool, NumRepInPool)
numRep2 = fun2(_pool, NumRepInPool)

Figure 4: SES with multi-aspect and specialization followed by multi-aspects and a possible PES.

The first and second pruning steps are the same as in Section 3.2, while in the third pruning step the attributes *numRep1* and *numRep2* are evaluated using SESfcn *fun2*. Depending on the value in attribute *_pool* and the SESvar *NumRepInPool* the number of *notebook* and *desktop* entities are generated. For evaluation of the SESfcn all attributes in the paths from *notebooksMASP* and *desktopsMASP* respectively to the root node need to be accessible.

3.4 Multi-Aspect with Succeeding Specialization and Sequenced Multi-Aspects

Now the SES is once again expanded by a new layer. Each entity *notebook* or *desktop* can be detailed with a different number of processor *core*s. As depicted in Figure 5, the *notebook* and *desktop* entities are therefore followed by multi-aspects *coresMASP*. Due to the SES axiom of uniformity, nodes with the same name need to have the same variables and isomorphic subtrees. Thus, the nodes *coresMASP* need to have the same *numRep* attribute in both specialized branches, here named *numRep3*. The value assignment to *numRep3* is variable, because the number of cores can be different for each desktop or notebook. Therefore it is set dynamically by the SESfcn *fun3*. An SESvar *NumCores* is passed as input argument. The *Semantic Condition* for *NumCores* defines that it is a list of vectors. The number of list elements is equal to the number of pools, defined by *NumPools*. The number of elements in each vector is equal to the number of entities in the corresponding pool, defined by *NumRepInPool(i)*.

The pruning process is described for the following SESvar values:

- NumPools = 2
- Types = ['nb', 'dt']
- NumRepInPool = [2, 3]
- NumCores = [[1, 2], [1, 2, 1]]

The first, second, and third pruning steps are known from previous sections. In the fourth pruning step the values of all *numRep3* attributes are calculated using the SESfcn *fun3*. The SESfcn uses a predefined function *path()*, which is a general graph function that returns a list of all node objects in a path from the calling node up to the root. Here, the SESfcn *fun3* is called at each node *coresMASP*. Using the *_pool* and *_notebook* or *_desktop* attributes of the objects in the path *fun3* determines the respective value in the SESvar *NumCores* and returns the number of cores.

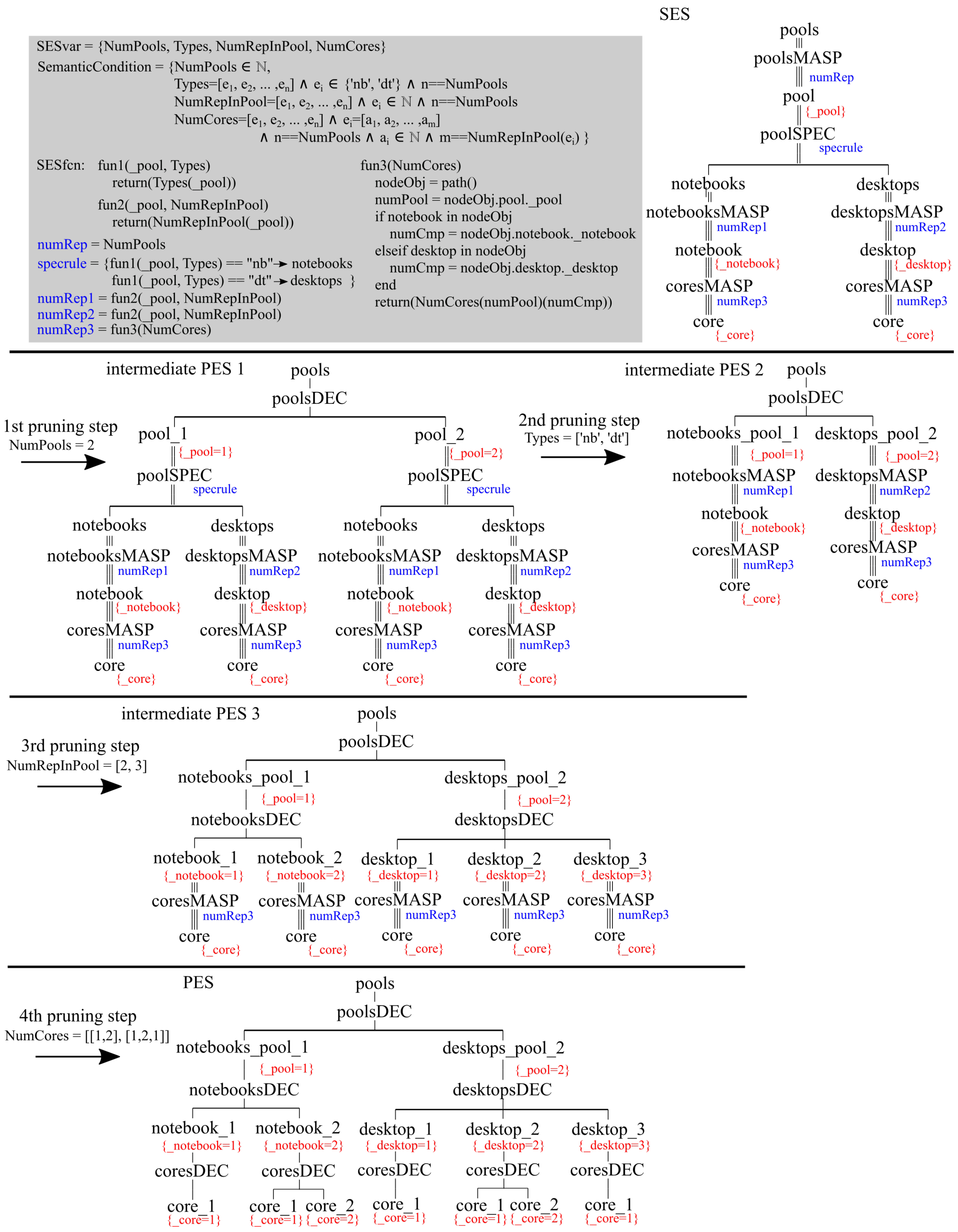
A fundamental prerequisite for the described approach for automated pruning of SES with hierarchical multi-aspects is the introduction of the additional underscore attribute at a generating entity. As can be seen in Figure 5, the axiom uniformity is no longer valid in the PES, because nodes with the same name have no isomorphic subtrees. However, the axiom can be relaxed in the context of the motivation shown in Section 2. Of course, one could have avoided some of the challenges discussed before for example by choosing different names for the nodes *coresMASP* in the different branches or by restructuring the SES. The disadvantage is then a more complex tree. Complex trees are beautiful in nature, but quickly confusing in modeling and simulation.

The suggested approach using the additional underscore attribute and the general path function can be applied to problems with deeper hierarchy, such as the SES depicted in Figure 6, in the same manner.

4 RELATED WORK

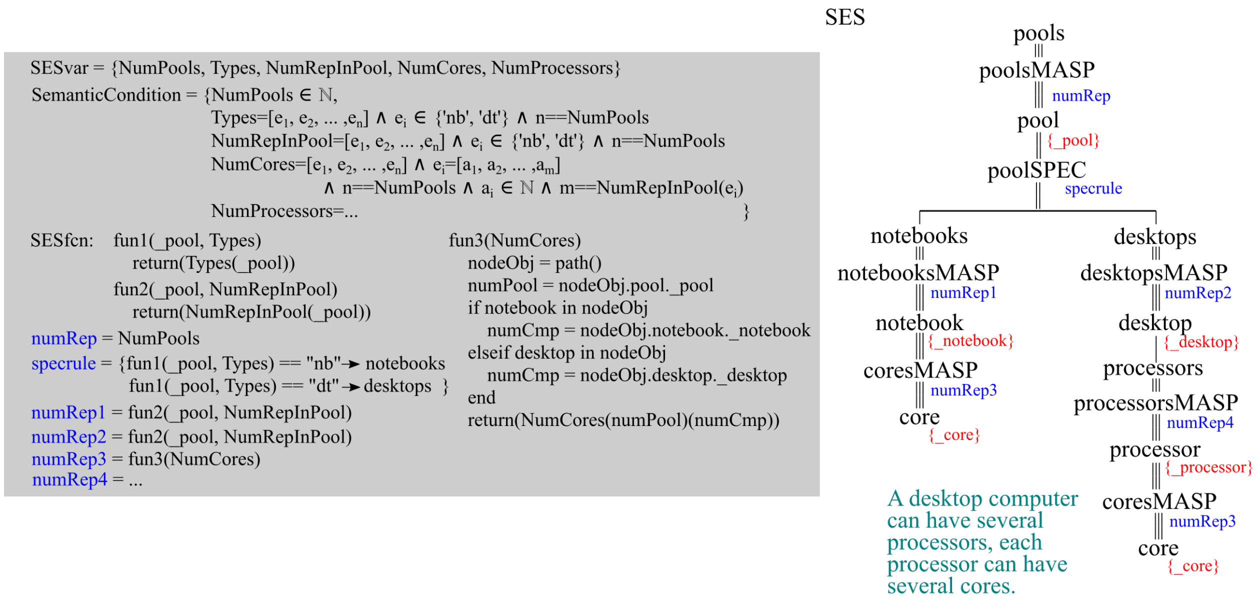
The research is motivated by current problems in the field of engineering, which is proven in Section 1 by selected works from the automotive sector. The method developed is based directly on SES/MB approach. Essential work on the state of research of SES/MB formalism is described and listed in Section 1. Besides the SES/MB approach that is derived from general system theory, a multitude of methods for variant modeling and variant generation have been developed in the field of software engineering. A good overview about these methods, some tools and experiences are given by Capilla, Bosch, and Kang (2013). In the context of engineering applications, feature-based modeling approaches, such as by Oster (2012) or Fang et al. (2016), are often used. Furthermore, the widely used UML has been extend with feature-based approaches to model variability of system families, such as by Robak, Franczyk, and Politowicz (2002).

The work in software engineering continuously influences the field of modeling and simulation. For describing system structures and their configurations, various XML based composition schemes, which distinguish between interfaces and the concrete implementations of the models were proposed, such as by Röhl and



```
SESvar = {NumPools, Types, NumRepInPool, NumCores}
SemanticCondition = {NumPools ∈ ℕ,
                     Types=[e_1, e_2, ... ,e_n] ∧ e_i ∈ {'nb', 'dt'} ∧ n==NumPools
                     NumRepInPool=[e_1, e_2, ... ,e_n] ∧ e_i ∈ ℕ ∧ n==NumPools
                     NumCores=[e_1, e_2, ... ,e_n] ∧ e_i=[a_1, a_2, ... ,a_m]
                                   ∧ n==NumPools ∧ a_i ∈ ℕ ∧ m==NumRepInPool(e_i) }

SESfcn:  fun1(_pool, Types)
           return(Types(_pool))

         fun2(_pool, NumRepInPool)
           return(NumRepInPool(_pool))

numRep = NumPools
specrule = {fun1(_pool, Types) == "nb"→ notebooks
            fun1(_pool, Types) == "dt"→ desktops  }
numRep1 = fun2(_pool, NumRepInPool)
numRep2 = fun2(_pool, NumRepInPool)
numRep3 = fun3(NumCores)

fun3(NumCores)
  nodeObj = path()
  numPool = nodeObj.pool._pool
  if notebook in nodeObj
     numCmp = nodeObj.notebook._notebook
  elseif desktop in nodeObj
     numCmp = nodeObj.desktop._desktop
  end
  return(NumCores(numPool)(numCmp))
```

Figure 5: SES with multi-aspect and specialization followed by sequenced multi-aspects and a possible PES.



```
SESvar = {NumPools, Types, NumRepInPool, NumCores, NumProcessors}
SemanticCondition = {NumPools ∈ ℕ,
                     Types=[$e_1$, $e_2$, ... ,$e_n$] ∧ $e_i$ ∈ {'nb', 'dt'} ∧ n==NumPools
                     NumRepInPool=[$e_1$, $e_2$, ... ,$e_n$] ∧ $e_i$ ∈ {'nb', 'dt'} ∧ n==NumPools
                     NumCores=[$e_1$, $e_2$, ... ,$e_n$] ∧ $e_i$=[$a_1$, $a_2$, ... ,$a_m$]
                              ∧ n==NumPools ∧ $a_i$ ∈ ℕ ∧ m==NumRepInPool($e_i$)
                     NumProcessors=...                                        }

SESfcn:  fun1(_pool, Types)                           fun3(NumCores)
           return(Types(_pool))                         nodeObj = path()
         fun2(_pool, NumRepInPool)                      numPool = nodeObj.pool._pool
           return(NumRepInPool(_pool))                  if notebook in nodeObj
numRep = NumPools                                         numCmp = nodeObj.notebook._notebook
specrule = {fun1(_pool, Types) == "nb"→ notebooks       elseif desktop in nodeObj
            fun1(_pool, Types) == "dt"→ desktops  }       numCmp = nodeObj.desktop._desktop
numRep1 = fun2(_pool, NumRepInPool)                     end
numRep2 = fun2(_pool, NumRepInPool)                     return(NumCores(numPool)(numCmp))
numRep3 = fun3(NumCores)
numRep4 = ...
```

Figure 6: SES with deep hierarchy of multi-aspects.

Uhrmacher (2006) or by Wang and Wainer (2015). Although that research does not explicitly address model families, some of the basic ideas are similar. Moreover, the ideas in Röhl and Uhrmacher (2006) and Wang and Wainer (2015) are important for the design of reusable model components and their organization in an MB. The relations between the SES formalism and XML are discussed by Zeigler and Hammonds (2007).

A common practical tool for today's modeling and simulation applications in engineering is MATLAB/Simulink (MathWorks 2019). Simulink as one of the most popular tools for model-based development provides special blocks for switching between model structures on the dynamic system level, the Variant Subsystems Blocks. This approach can be combined with MATLAB's Variant Manager or third party products, such as pure::variants (Pure Systems 2019), which are specialized for variant management. Anyhow, all system variants need to be coded on the low dynamic system level. In contrast, the SES/MB approach offers the possibility to define "growing" systems in a structured, clearly arranged way. A comparative evaluation of both approaches is given by Deatcu, Pawletta, and Folkerts (2019).

5 CONCLUSION

This paper deals with challenges of using SES for modeling specific families of systems and deploy the SES in frameworks for automated, reactive simulation experiments. For efficient variability modeling, the considered problems require the usage of multi-aspect and specialization nodes in a deep hierarchical manner. Extensions of the SES for goal-directed automated pruning of such SES trees are introduced. The modeling of SES and the pruning process are illustrated using an artificial example, which is extended step by step. The case study clarifies that the discussed approach is applicable to such SES. However, with increasing hierarchy depth of multi-aspects and specializations, the specification of the introduced SES variables can become complicated and thus error-prone.

In simulation engineering, modeling of versatile systems and their automated, goal-directed study is a challenging task. This work opens the perspective to solve such problems and to implement appropriate software frameworks as also discussed in the paper. Future works will especially cover the usage of the presented methods in real engineering applications.

REFERENCES

Capilla, R., J. Bosch, and K. C. Kang. (Eds.) 2013. *Systems and Software Variability Management, Concepts, Tools and Experiences*. Springer.

Deatcu, C., H. Folkerts, T. Pawletta, and U. Durak. 2018. "Design Patterns for Variability Modeling Using SES Ontology". In *Proceedings of the Model-driven Approaches for Simulation Engineering Symposium*, Mod4Sim '18, pp. 3:1–3:12. San Diego, CA, USA, Society for Computer Simulation International.

Deatcu, C., T. Pawletta, and H. Folkerts. 2019, March. "MATLAB/Simulink's Variant Manager vs SES-ToPy". *SNE Simulation Notes Europe* vol. 29 (1), pp. 39–43.

Fang, M., G. Leyh, J. Dörr, and C. Elsner. 2016. "Multi-variability modeling and realization for software derivation in industrial automation management". In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*, edited by B. Baudry and B. Combemale, pp. 2–12, ACM.

Folkerts, H., T. Pawletta, C. Deatcu, and S. Hartmann. 2019, December. "Python-based eSES/MB Framework: Model Specification and Automatic Model Generation for Multiple Simulators". *SNE Simulation Notes Europe* vol. 29 (4), pp. 207–215.

Grönniger, H., H. Krahn, C. Pinkernell, and B. Rumpe. 2014. "Modeling Variants of Automotive Systems using Views". *CoRR* vol. abs/1409.6629.

Maria, A. 1997. "Introduction to Modeling and Simulation". In *Proceedings of the 29th conference on Winter simulation, WSC 1997, Atlanta, GA, USA, December 7-10, 1997*, edited by S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, pp. 7–13, ACM.

MathWorks 2019. https://www.mathworks.com/. Accessed Dec. 12, 2019.

Oster, S. 2012, Januar. *Feature Model-based Software Product Line Testing*. Ph. D. thesis, Technische Universität, Darmstadt.

Pawletta, T., A. Schmidt, U. Durak, and B. P. Zeigler. 2018, December. "A Framework for the Metamodeling of Multivariant Systems and Reactive Simulation Model Generation and Execution". *SNE Simulation Notes Europe* vol. 28 (1), pp. 11–18.

Pawletta, T., A. Schmidt, B. P. Zeigler, and U. Durak. 2016. "Extended Variability Modeling Using System Entity Structure Ontology Within MATLAB/Simulink". In *Proceedings of the 49th Annual Simulation Symposium*, ANSS '16, pp. 22:1–22:8. San Diego, CA, USA, Society for Computer Simulation International.

Pure Systems 2019. https://www.pure-systems.com/. Accessed Dec. 12, 2019.

Robak, S., B. Franczyk, and K. Politowicz. 2002. "Extending the UML for Modelling Variability for System Families".

Rozenblit, J. W., and Y. Huang. 1991. "Rule-Based Generation of Model Structures in Multifaceted Modeling and System Design". *INFORMS Journal on Computing* vol. 3 (4), pp. 330–344.

Rozenblit, J. W., and B. P. Zeigler. 1993. "Representing and Constructing System Specifications Using the System Entity Structure Concepts". In *Proc. of the 25th Conference on Winter Simulation*, WSC '93, pp. 604–611. New York, NY, USA, ACM.

Röhl, M., and A. M. Uhrmacher. 2006. "Composing simulations from XML-specified model components". In *Proceedings of the 2006 Winter Simulation Conference*, pp. 1083–1090.

Santucci, J. F., L. Capocchi, and B. P. Zeigler. 2016. "System entity structure extension to integrate abstraction hierarchies and time granularity into DEVS modeling and simulation". *Simulation* vol. 92 (8), pp. 747–769.

Schmidt, A. 2019. *Variant Management in Modeling and Simulation Using the SES/MB Framework*. Ph. D. thesis, Rostock University.

Schmidt, A., U. Durak, and T. Pawletta. 2016. “Model-based Testing Methodology Using System Entity Structures for MATLAB/Simulink Models”. *SIMULATION* vol. 92 (8), pp. 729–746.

Wang, S., and G. A. Wainer. 2015. “Semantic selection for model composition using SAMSaaS”. In *Proceedings of the 2015 Spring Simulation Conference*, TMS-DEVS ’15, pp. 25–32.

Zeigler, B., T. Kim, and H. Praehofer. 2000. *Theory of Modeling and Simulation*. Elsevier Science.

Zeigler, B. P. 1984. *Multifacetted Modelling and Discrete Event Simulation*. San Diego, CA, USA, Academic Press Professional, Inc.

Zeigler, B. P., and P. E. Hammonds. 2007. *Modeling & Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange*. Orlando, FL, USA, Academic Press, Inc.

Zeigler, B. P., and H. S. Sarjoughian. 2013. *Guide to Modeling and Simulation of Systems of Systems*. Simulation Foundations, Methods and Applications. Springer.

AUTHOR BIOGRAPHIES

HENDRIK FOLKERTS is a PhD student at Wismar University, Germany. He is developing a tool for creating an SES including pruning and model generation for different simulation tools. His email address is hendrik.folkerts@cea-wismar.de.

THORSTEN PAWLETTA is a Professor for Applied Computer Science at Wismar University, Germany. His research interests lie in M&S theory and application in engineering. His email address is thorsten.pawletta@hs-wismar.de.

CHRISTINA DEATCU is a research engineer at Wismar University, Germany. Her research interests lie in theory of modeling and simulation, especially of discrete event and hybrid systems. Her email address is christina.deatcu@hs-wismar.de.

BERNARD P. ZEIGLER is Chief Scientist at RTSync Corp., Professor Emeritus of Electrical and Computer Engineering at the University of Arizona, Tucson and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His email address is zeigler@rtsync.com.