Ontology-Assisted System Modeling and Simulation within MATLAB/Simulink

Thorsten Pawletta*, Daniel Pascheka, Artur Schmidt, Sven Pawletta

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

Simulation Notes Europe SNE 24(2), 2014, 59 - 68
DOI: 10.11128/sne.24.tn.10241
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.),
Revised: Oct. 20, 2014; Accepted: October 30, 2014

Abstract. Ontology-assisted system modeling combines classic system-theoretical modeling with an ontological system specification. Different dynamic system behavior is modeled in configurable basic models with defined input and output interfaces. Basic models are organized in a model base (MB). The ontology is used to specify a set of modular, hierarchical system structures using references to basic models in the MB. Moreover, the ontological model defines possible parameter settings of referenced basic models. Thus, the ontology describes a set of different system configurations for a specific domain. A base ontology for mapping such problems is the System Entity Structure (SES). A combination of SES ontology with a MB for system modeling and goal-oriented model generation was introduced with the SES/MB framework.

Starting with the basics of SES ontology and SES/MB framework as well as the discussion of some extensions, a new SES toolbox for ontological modeling within the MATLAB/Simulink environment is presented. The toolbox architecture is then discussed. The main focus in this regard is on the graphical SES editor, the toolbox methods and the seamless integration with MATLAB/Simulink. The latter is described by means of deriving a specific system model from the formal specification and the automatic generation of a corresponding executable MATLAB/ Simulink model.

Introduction

Current simulation environments support modular, hierarchical modelling and the combination of different modeling formalisms, and provide powerful numerical methods for simulation and data evaluation.

The conceptual modeling phase and data modeling according to the lifecycle model in [1], as well as experiment descriptions of various system models and data sets or a combination with other numerical methods, are not yet considered equivalently.

Experimentation with different system designs or variants is a requirement that is becoming increasingly more important. Usually, all system variants have to be modeled as separate dynamic system models and their investigation is carried out manually or via experiment scripts.

Some simulation environments, such as MATLAB/ Simulink, support variant modeling on the level of dynamic system models by using component-based techniques. The activation of a certain variant is carried out using specific control variables [17], which are defined in the system model. This allows simplified experimentation with a limited set of variants. Sometimes, this approach is combined with external tools for variant modeling [6] [7]. Then, the challenge is the synchronization of the external variant model with the dynamic system models.

The ontology-assisted modeling intends a more holistic approach that supports the process of modeling and simulation from the conceptual phase to goal-oriented experimentation with various system variants. The term ontology originates from philosophy and means theory of existence. In computer science ontology is basically defined as a formal structured representation of concepts and their relations. However, ontology is often employed differently and contradictorily in computer science [5]. In the following, ontology is used as defined in [5] [16] [3]. Thus, ontology is understood as a formal specification of a shared conceptualization in the form of a model with a ‘closed world assumption’. The latter denotes that true is only what is explicitly specified in the model.

According to [20], the considered domain of conceptualization is modeling and simulation of modular, hierarchical systems. In this context, ontology-assisted characterizes a declarative specification of various system structures and parameter settings in combination with configurable basic models. Basic models map different dynamic system behavior, define an input and output interface and are organized in a model base (MB). The ontology specifies references to basic models and defines admissible parameter settings for them. Similarly, ontology can be used to specify a set of different experiments with the system models. In this case, the ontological specification describes the composition of experiments using references to various experiment methods or data, such as employed in [12] for model-based testing [18]. The experiment methods or data are organized in an MB or data base analogous to basic models. Because of its declarative character an ontological specification can be utilizsed in the early phases of the lifecycle model, e.g. during conceptual or data modeling, and can be extended stepwise.

Zeigler, et al. developed the *System Entity Structure* (SES), a base ontology for system and data modelling [19][21]. Based on the SES ontology they derived the SES/MB framework [20]. The framework combines an SES with an MB and proposes basic methods for deriving a concrete system model and for generating an executable simulation model. A software implementation of the SES/MB framework is presented in [22] and called MS4Me. MS4Me is implemented in JAVA and based on the *Discrete Event System* (DEVS) formalism according to [19][20]. That means, basic models have to be specified according to the DEVS formalism.

The research in [4][11][13] shows that the concept of SES/MB is well suitable for solving complex engineering problems. The SES ontology is based on a clear, limited set of description elements and axioms. Thus, it is more easily usable for engineers than alternative developments such as Protegè [10]. However, an important precondition for the application of new concepts in engineering is their availability in an engineering software environment and their direct combination with established methods. MS4Me does not comply with these conditions. For this reason, an earlier toolbox, called Tiny SES toolbox, was implemented for the well accepted MATLAB/Simulink environment [14][15]. Use of this toolbox requires basic knowledge of first-order logic and the connection of a PROLOG interpreter to MATLAB. Both things are often daunting for engineers.

Based on the Tiny SES toolbox a new and extended toolbox for MATLAB/Simulink has been developed. It is completely implemented and integrated in MATLAB, requires no deeper understanding of first-order logic and provides a graphical front-end for SES-based modeling. In addition, it provides different methods to derive system models from an SES and to generate executable simulation models for Simulink using predefined blocksets or subsystems. In the same way models for SimEvents or the MATLAB/DEVS Toolbox [2] can be generated automatically with little effort.

The basics of SES/MB framework and originary SES ontology, as well as new introduced features, are first described. Then, the toolbox architecture and provided methods are discussed. Finally, a summary and a look forward to future work are given.

1 Theoretical Backgrounds

The pragmatic research presented in this paper is based on the long-term theoretical works of Zeigler, et al. [20][21]. In the following, the conceptional *System Entity Structure and Model Base* (SES/MB) framework and fundamental ideas of the underlying SES ontology are summarized. Moreover, restrictions and extensions of the SES, related to the toolbox development that is described in the next section, are discussed. Subsequently, the pruning process to derive a distinct system configuration from an SES is considered.

1.1 SES/MB Framework

The SES/MB framework introduced by Zeigler et al. in [20] combines the SES ontology with the classical approach of modeling and simulation of modular-hierarchical systems. Figure 1 illustrates the principle elements and operations.

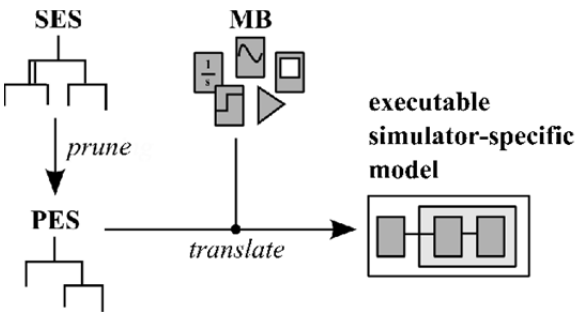


Figure 1: SES/MB framework according to [20].

Configurable basic models with a defined input and output interface are organized in an MB. They describe different dynamic behavior. The SES is a special kind of tree structure. It describes a set of possible system structures for a closed domain. To do so, it specifies references to basic models in the MB and defines possible parameter settings for them. In addition, an SES can specify a set of goal-directed experiments, but this is not taken into consideration. Hence, the SES can be considered as a variable construction plan for different system configurations or variants. The selection of a specific system configuration is based on a pruning operation. The result of pruning is a tree structure that describes a unique system configuration and is called *Pruned Entity Structure* (PES). Based on the information of PES, and using models from the MB, an executable simulation model can be generated via an appropriate translator.

1.2 Originaly SES ontology and modifications

The SES ontology is based on a directed and labeled tree. It defines different types of nodes and edges as well as a set of axioms. They are summarized in Figure 2 with respect to their category and affiliation.

```
mSES:
  ELEMENTS:
    NODES:
      Entity
        Attributes
      DESCRIPTIVE NODES:
      Aspect
     -MultiAspect
      Specialization
    EDGES:
      Entity Edge
       +Selection Rules of Aspect Siblings
      Aspect Edge
        Couplings
      Specialization Edge
        Selection Rules
      MultiAspect Edge
        Replication Var. & Couplings
       +Selection Constraints
    SEMANTIC RELATIONS
   +SES VARIABLES, FUNCTIONS, PRIORITIES
  AXIOMS:
    Alternate Mode
    Strict Hierarchy
    Uniformity
    Valid Brothers
    Assigned Attributes (Variables)
    Inheritance
```

Figure 2: Elements and Axioms of mSES.

In the context of toolbox implementation some restrictions and extensions compared with the originary SES definition in [21] are introduced. Extended or new elements are marked with a beginning plus sign and elements with restrictions with a beginning minus sign. The term mSES (modified SES) is used to distinguish from the originary definition. However, the term SES is also still used in regards to linguistic simplification.

On the basis of a fictitious, arresting example, the basic elements and axioms will be explained. The scope of the subject is the conceptualization of melt behavior of different structured ice cream portions (**Ip**), as illustrated in Figure 3.

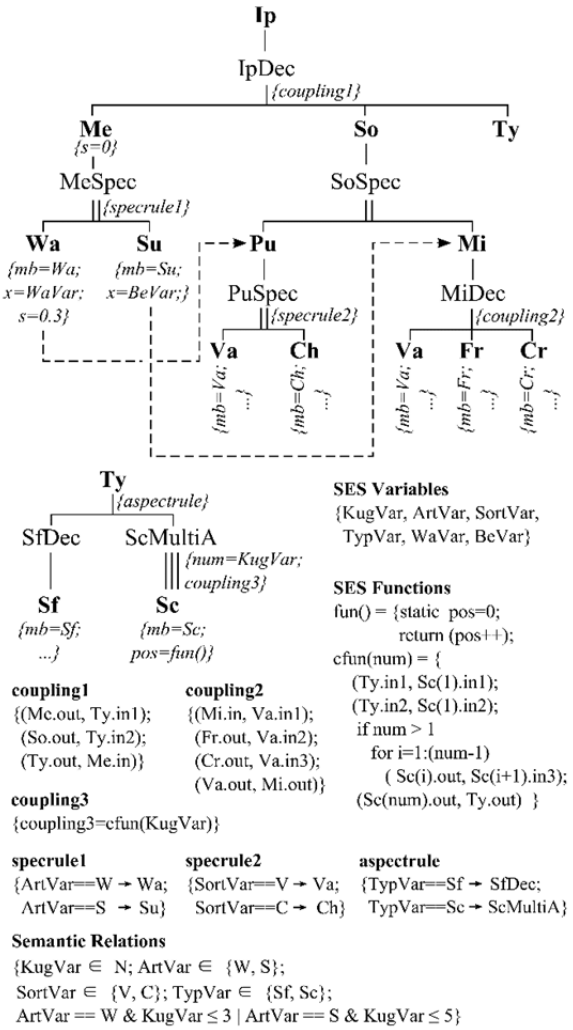


Figure 3: mSES for the ice cream portion example.

The mSES is partitioned in two trees, which are merged via the ***Ty*** node. Details of merging SESs are explained in the next section. The composition of an ***Ip*** is based on the following considerations:

```
Operators: []compose, |xor, =is, ()abbr
Ip = [ Medium(Me), Sort(So), Type(Ty) ]
Me =  Wafer(Wa) | Sundae(Su)
So =  Pure(Pu)  | Mixed(Mi)
    Pu = Vanilla(Va) | Choco(Ch)
    Mi = [ Va, Fruit(Fr), Cream(Cr) ]
Ty = Soft(Sf) | Scoop(Sc)
    Sc = [ once | twice | … | n_times ]
```

Nodes pictured in bold are *entities*, the others are *descriptive nodes*. Both node types alternate due to the *1st axiom* in Figure 2. The *leaf nodes* map atomic entities, which define in their special node attribute *mb* a reference to a basic model in the MB. In this case the MB contains the following types of basic models: *MB={Wa, Su, Va, Ch, Fr, Cr, Sf, Sc}*. The names of leaf nodes need not be the same as the names of the basic models, even though they are in the example.

Descriptive nodes characterize their predecessor node. The suffixes denote: *Dec* ≡ *Aspect*, *MultiA* ≡ *MultiAspect*, *Spec* ≡ *Specialization*. In the ice cream portion example, the Aspects *IpDec*, *MiDec*, *SfDec* describe a decomposition of their predecessor node. That means their predecessor entity node represents a composed system on the base of their successor entity nodes respectively, whereby a composition can also consist of a single entity. In addition, an *Aspect edge* can specify *couplings* for a composition as an attribute in the form of *(from, to) relations* (see definitions of *coupling1*, *coupling2* in Figure 3).

Analogously, the MultiAspect *ScMultiA* describes a composition of its predecessor entity node based on the number of entities of the same type (replications) defined by its successor node *Sc*. Moreover, each *MultiAspect edge* defines a range for the valid number of replications in its attribute *num*. The coupling relations can depend on the number of replications, such as in Figure 3 in attribute *coupling3*. Such kinds of variability can be easily specified with the newly introduced *SES Functions*. Its usage is described at the end of this subsection.

An entity node can define several *Aspect* or *MultiAspect* nodes as successor nodes, which are called *sibling nodes*. In Figure 3, the successor nodes *SfDec* and *ScMultiA* of entity node *Ty* are such sibling nodes.

With respect to the SES toolbox, in such cases the *entity edge* can be defined as attribute *Selection Rules of Aspect Siblings*. In the case of node *Ty*, this is done by the edge attribute {*aspectrule*}. Alternatively, a selection can also be defined using *Selection Constraints*, which are represented as broken lines.

Specialization nodes like *MeSpec*, *SoSpec* or *PuSpec* describe the taxonomy of their predecessor node. This means their superior entity node is only an abstraction with respect to their subsequent entity nodes. The conditions for selecting a successor node are specified with *Selection Rules* on the *specialization edge*, or with *Selection Constraints*, analogous to the selection of sibling nodes previously described (see Figure 3 selection constraints between *Wa – Pu* and *Su – Mi* as well as selection rules in attributes *specrule1* and *specrule2*). The specialization relation (taxonomy) between entity nodes is based on the powerful *inheritance axiom* that defines a unification of an abstract father entity node with a selected child entity node, regarding the *node name*, *attributes* and *subtrees*. Some effects of this axiom are described in the next subsection.

In addition, the inheritance axiom can cause side effects, if subtrees are inherited. To guarantee a unique specification, *Priorities* are introduced as an additional SES element. A detailed description of side effects and their avoidance using *Priorities* is described in [8]. The mSES in Figure 3 contains no inheritance of subtrees.

Node and edge attributes can define constant or variable expressions. The known concept of *SES Variables* has been extended by *SES Functions*. The mSES in Figure 3 defines the two SES Functions *fun*() and *cfun*(*num*). The first one is used for defining the position of scoops in an ice cream portion (Figure 3, attribute *pos* of node *Sc*) and the second one for specifying the coupling relations between scoops depending on the number of scoops in an ice cream portion (Figure 3, attribute *coupling3* of node *ScMultiA*). Moreover, an mSES can specify *Sematic Relations*. The evaluation of *SES Variables* and *SES Functions*, as well as the examination of *Semantic Relations* are explained in the next subsection. With respect to the developed SES toolbox it should be mentioned, that SES Functions can be coded in pure MATLAB syntax using built-in MATLAB functions. The pseudocode in Figure 3 is used for simplification.

It can be concluded that the mSES in Figure 3 specifies 14 valid compositions of an ice cream portion (**Ip**).

1.3 Deriving a PES – T

1.4 the Pruning Operation

For deriving a unique, valid system configuration Zeigler et al. defined in [20] a pruning operation for an SES. The result of pruning is a tree, called *Pruned Entity Structure* (PES). The PES is a unique tree without decision points and variable attributes, which means all variabilities are resolved. The basic ideas of pruning are described step by step by means of the mSES in Figure 3.

Before pruning, all SES Variables have unique values assigned. Let's assume the following assignments:

```
ArtVar=W for wafer; SortVar=V for vanilla;
TypVar=Sc for scoop; KugVar=3 for #scoops;
WaVar=9.5 for Wafer parameter x; BeVar=Ø
```

The pruning operation is based on a depth-first search. With respect to the mSES in Figure 3, it starts at root node *Ip* with its subsequent Aspect node *IpDec* and edge attribute *{coupling1}* as well as its follower nodes *Me*, *So* and *Ty*. This subtree contains no decisions and that is why it is copied without change to the PES. After that, the Specialization node *MeSpec* with its edge attribute *{specrule1}* is evaluated and the entity node *Wa* is selected. Based on the inheritance axiom, the entity nodes *Me* and *Wa* will be fused. The result is a new entity node, *Wa_Me*.

In the PES, the node *Me* is replaced by the new node *Wa_Me* with the attributes *{mb=Wa; x=9.5; s=0.3}*. With respect to this, the coupling attribute *{coupling1}* of *IpDec* needs to be updated in its first and third coupling relation (see Figure 4). Furthermore, the *Selection Constraint* between the nodes *Wa* and *Pu* indicates the selection of *Pu*. Now, the subtree of *Me* is fully analyzed and the pruning is continued at node *So*. Because of the pre-decided selection of entity node *Pu*, the Specialization node *SoSpec* leads to the unification of the entity nodes *So* and *Pu* into a new entity node, *So_Pu*. In the PES, the node *So* is replaced by the new node *Pu_So*. The following analysis of *PuSpec* with its edge attribute *{specrule2}* results in the selection of node *Va*. In the same manner the new node *Pu_So* is fused with node *Va* and its attributes. The result is a node *Va_Pu_So* with the attributes *{mb=Va;…}*. According to this operation, in the PES, the edge attribute *{coupling1}* of *IpDec*, needs to be updated in the second coupling relation (see Figure 4). Pruning is now continued with the entity node *Ty*, where the edge attribute *{aspectrule}* leads to the selection of the MultiAspect node *ScMultiA*.

Based on the edge attribute *{num=KugVar}* and the SES Variable assignment *KugVar=3*, three entities of type *Sc* are generated. According to the second edge attribute *{coupling3}*, the coupling relations of the three new nodes are calculated by the SES Function *cfun(num)* with the actual value assignment *num=3*. Due to the *valid brothers* axiom the generated entities are renamed using consecutive numbers, whereby the entity nodes *Sc1*, *Sc2*, *Sc3* are created for the PES.

Regarding this, the coupling relations in attribute *{coupling3}* are also renamed in the PES (see fig. 4). Each of them has a fixed node attribute *{mb=Sc}*. The variable node attribute *{pos=fun()}* is separately calculated for each node using the SES Function *fun()*. The *pos* attribute describes the position of the scoop in the ice cream portion *Ip*. After completing this procedure, a complete PES, as depicted in Figure 4, is derived.

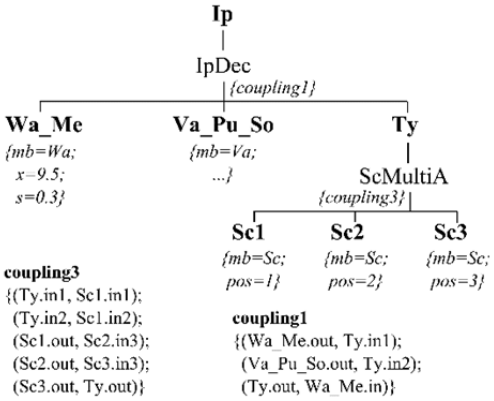
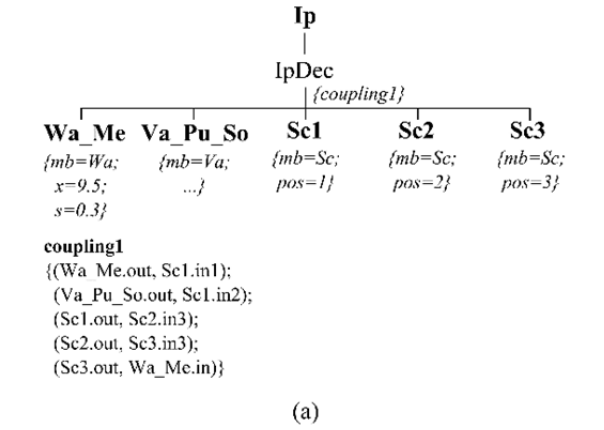


Figure 4: A unique, valid PES of the mSES in Figure 3.

Finally the validity of the PES needs to be proved by evaluating the *Semantic Relations* using a logical AND operation. The PES pictured in Figure 4 is valid and maps a unique system configuration.

According to the attribute *{coupling3}* in Figure 4, the entity node *Ty* describes a composed system. From the perspective of system dynamics, it can be resolved. The resolution of composed systems reduces system complexity.

In context of pruning, this is a called flattening and such a reduced PES is called *Flatted Pruned Entity Structure* (FPES). Flattening requires, in this case, a modification of coupling relations in attribute *{coupling1}*. Figure 5a shows the resulting FPES and Figure 5b the corresponding system structure.



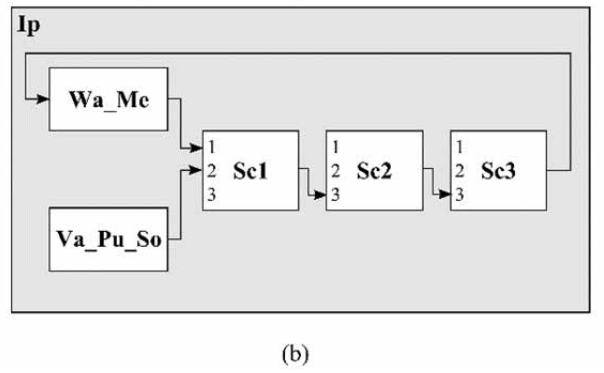


Figure 5: (a) FPES of PES in Fig. 4; (b) corresponding system structure.

On the basis of PES or FPES, an executable simulation model can be generated using basic models from the MB, if an appropriate translator is available (see Figure 1). At this point, it should be mentioned that the identifiers in Figure 5b represent the names of system components.

The names of associated basic models are coded in the particular node attribute *{mb}*.

To conclude, it is noted that the pruning operation of the SES toolbox discussed below is restricted to *Multi-Aspect* nodes, with a subsequent entity node that has to be a leaf node.

2 SES Toolbox for MATLAB/Simulink

Various research in [4][11][13] show that the concept of SES/MB is well suitable for solving complex engineering problems, if it is available in an engineering software environment. In the following, the fundamental aspects of a new SES toolbox for MATLAB/Simulink are presented. Beginning with a description of the software architecture, the basic methods of the toolbox are discussed.

2.1 Software Architecture and User Interface

Figure 6 shows the software architecture of the toolbox in the form of a UML class diagram.

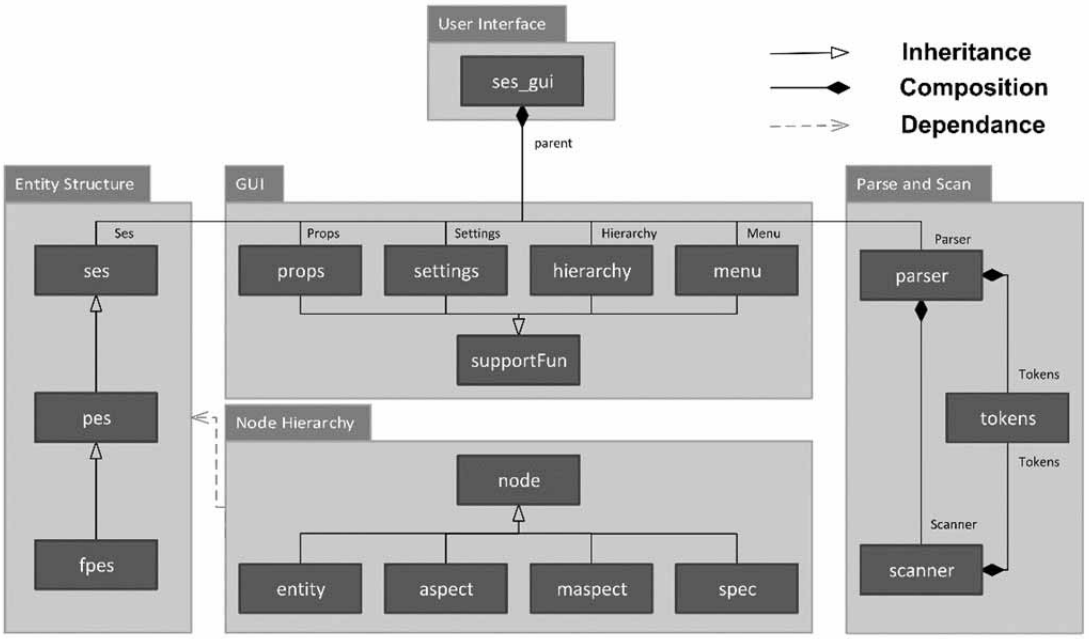


Figure 6. Class structure of SES toolbox.

Figure 7. Graphical User Interface of SES toolbox.

The classes are divided into five packages. The class *ses_gui* implements the user interface and constitutes the central interface class. The package GUI contains all classes that are necessary for the design and structure of the user frontend, as shown in Figure 7.

The GUI consists of a menu bar and three subwindows: *(i) Node Properties*, *(ii) Model Hierarchy* and *(iii) Global Settings*. All GUI classes are derived from a common superclass. In the subwindow *ModelHierarchy* a SES tree can be edited in a similar manner to a data manager. Selection Constraints are highlighted using different colors. Node and edge attributes are edited and displayed in the subwindow, *Node Properties*. The global properties of an SES, such as SES Variables, SES Functions and Semantic Relations, are managed in the subwindow, *Global Settings*. All user-related methods are provided via the menu bar.

Data structures and methods for the internal storage and management of an SES are defined in the classes of the packages *Entity Structure* and *Node Hierarchy*. Furthermore, they define methods for pruning an SES, flattening a PES and the merging of SESs. These are described in more detail in the next subsection.

The PES and FPES are considered as specializations of an SES and are, therefore, defined as subclasses of *ses class*. Thus, a PES or FPES can be managed and displayed with all its information using the GUI analogously to an SES.

The package *Parse and Scan* contains classes implementing a parser for lexical and syntactical analysis. The parser continuously checks all user inputs for their validity. In addition to the syntax checks of user inputs, the parser also performs semantic checks based on already saved information in order to ensure consistency.

2.2 Methods: Merging, Pruning, Flattening

Based on the fundamentals of the SES/MB framework and the SES ontology the toolbox provides methods for merging SESs and for pruning a SES as well as flattening a PES. These methods can be accessed in the GUI via the menu bar or as usual MATLAB functions.

The *merging* method supports the concatenation of an SES from various SESs, analogous to Figure 3. Each SES can define its own global settings, such as SES Variables, SES Functions or Semantic Relations. For merging one SES has to be qualified as the main SES. Code 1 shows the basic steps of the merging method.

```
load main SES; select merge node;
load imported SES;
if merging ~admissible -> Error;
for each leaf node homonymous with merge node
  merge imported SES tree to main SES tree;
merge global settings;
update displays in GUI
```

Code 1. Basic steps of merging method.

The selected merge node in the main SES has to be a leaf node of type entity. A merging operation is always applied to all leaf nodes homonymous to the selected merge node. The admissibility of merging is proved considering the SES axioms. During the merge process the root node name of imported SES is replaced by the name of the leaf node of the main SES.

Finally, the global settings of merged SESs are fused. Name conflicts will be resolved automatically in accordance with the SES axioms.

The toolbox provides three consecutive methods for *pruning* an SES and *flattening* a PES. These are: *(i) First-Level Pruning, (ii) Complete Pruning* and *(iii) Complete Pruning & Flattening*. Code 2 shows the basic steps of the consecutive methods.

```
FIRST-LEVEL PRUNING:
1. check pruning permission of SES;
2. verify SES Variables;
3. create SES Functions in MATLAB;
4. compute SES Vars which use SES Fcn;
5. transform Selection Constraints to
   Selection Rules;
6. depth-first search pruning;
7. check Semantic Relations;
        ->valid PES | interim_PES
8. if ~interim_PES
        set PES valid; ->END
COMPLETE PRUNING:
9. detect undecidable Aspect nodes & transform
   SES Priorities to Selection Rules;
10. depth-first search pruning;
11. check Semantic Relations;
        -> PES
12. if ~flattening
        set PES valid; ->END
FLATTENING:
13. rename homonymous leaf nodes;
14. depth-first search flattening;
15. set FPES valid; ->END
```

Code 2. Basic steps of pruning and flattening methods.

An SES gets pruning permission (1) when it satisfies the axioms. The verification of SES Variables (2) is only related to explicit method calls from the MATLAB prompt. In the case of method calls from the GUI, SES Variables are checked by the previously described parser.

The creation of SES Functions in MATLAB (3) needs to be executed uniquely for all SES Functions that are not defined as MATLAB built-in functions. Because an SES is saved as a data structure, SES Functions are encoded as strings, although they are edited as ordinary MATLAB Functions.

In step (4), SES Variables that depend on SES Functions are calculated. For simplification of the depth-first pruning (6) operation, which has been explained in section 2, in step (5) Selection Constraints are transformed into Selection Rules. Steps (7) and (8) verify whether, the resulting PES is complete and valid.

Due to the inheritance axiom, the subtrees of the parent node and the selected child node, at a specialization node, are combined as described in Subsection 1.2. This can cause undecidable Aspect nodes, although all SES axioms are considered. The *First-Level Pruning* (1-8) method enables the location of such nodes. The *Complete Pruning (9-12)* method uses the SES Priorities, introduced in [8] to resolve such nodes. The *Flattening (13-15)* method requires a complete, valid PES and creates a FPES according to the statements in Subsection 1.3.

2.3 Problem-oriented Model Translation

As shown in Section 1 an executable simulation model can be generated based on a PES or FPES using basic models from the MB, if an appropriate target translator is available. At present, the SES toolbox does not contain a general target translator for Simulink. It provides an M-file template that has to be adapted by the user. The general translation procedure is always the same. However, the parameter configuration of the various Simulink blocks is very different.

Up to now, the translation script has supported only a subset of the current Simulink blocksets. That is why we call the current translator, problem-oriented, because it has to be extended if new blocks are used. To minimize translation and model execution time, the translator is based on an FPES. Moreover, in most cases the structure of an executable model is of no relevance. The basic translation procedure, independent of a specific target, is depicted in Code 3. The basic translation steps for generating a Simulink model are then discussed.

```
If FPES is ~valid -> Error
INITIALIZATION:
 Instantiate empty model
 optional: set solver parameters
TRANSLATION:
 for each leaf node instance MATLAB obj.
 for each MATLAB obj. instance model obj.
     from the MB
 from Aspect attrib. instance model coupl.
FINALIZATION:
  optional: e.g. start simulation
```

Code3. Basic steps of model translation and execution.

Firstly, the validity property of the FPES that had to be set by the pruning method is checked. In the *initialization phase* a new, empty model is created. Its name is derived from the root node of FPES (see fig. 5(a)). Optionally, solver parameters and others can be adjusted. Solver adjustments depending on system variants can also be specified within the SES, which means they will also be encoded in the FPES. In this case, they will be generated in the *translation phase*.

Basically, solver parameters belong to an experiment specification, which should be clearly separated from the model specification. The real *translation phase* consists of three steps. In the first step, a MATLAB object is created for each leaf node of FPES. It stores all information of the leaf node, separated using the criteria: (i) node name, (ii) special attribute *mb* and (iii) remaining attributes. The special attribute *mb* stores the reference to a basic model in the MB; here, a Simulink block or subsystem.

In the next step a model object (here, a Simulink object) is instantiated and configured using the information in the MATLAB object. In the third translation step, the coupling relations, defined in the attribute of the Aspect edge (see fig. 5(a)), are evaluated and, according to this, the couplings of the target model are generated. It should be recalled that an FPES contains only one Aspect node. The *finalization phase* is optional.

Examples of engineering applications using the automatic generation of executable Simulink models from a SES specification can be found in [15][11][8][9].

3 Summary

The SES toolbox provides a comprehensive and user-friendly tool for ontological modeling in the MATLAB/Simulink environment. The toolbox is fully integrated in MATLAB/Simulink and can be used in seamless combination with other toolboxes and blocksets.

For the important domain of model-based system development the toolbox offers new ways of variant modeling within MATLAB/Simulink. The basic procedure has been demonstrated by means of a Simulink example. Current working priorities are the removal of existing restrictions when using the *Multi-Aspect* element, the development of further examples using other MATLAB toolboxes, such as Stateflow or Simscape, and a more general target translator for Simulink.

The toolbox can be accessed for free by registering at *http://www.mb.hs-wismar.de/cea/sw_projects.html*.

Acknowledgment. This work was partly supported by the German Research Foundation (DFG) under code number PA 631/2-2.

References

[1] O. Balci. *Verification, validation and testing.* In: J. Banks, ed., Handbook of Simulation, John Wiley & Sons Inc., 1998, 335 – 393.

[2] C.Deatcu, T. Schwatinski, T. Pawletta. DEVS Toolbox for MATLAB – MatlabDEVS Tbx., 2013, [Online] http://www.mb.hs-wismar.de/cea/DEVS_Tbx/MatlabDEVS_Tbx.html.

[3] T.R. Gruber. *A translation approach to portable ontology specifications.* Knowledge Aquisition, Vol. 5(1993)2, 199 – 220.

[4] O. Hagendorf, T. Pawletta, R. Larek. *An approach to simulation based parameter and structure optimization of MATLAB/Simulink models using evolutionary algorithms.* In: SIMULATION – Trans. of the Soc. for Modeling and Simulation Int., Vol. 89(2013)9, 1115 – 1127.

[5] B. Henderson-Sellers. *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages.* Springer Pub., 2012.

[6] H. Holdenschick, W. Commerell. *Variantenmanagement in der modellbasierten Produktentwicklung von Fahrzeugsystemen. (Variant Management in Model Based Produkt Development of Automotive Systems von Fahrzeugsystemen. In German) In: Proc. ASIM Meeting STS/GMMS, Düsseldorf, ARGESIM Report 41 / AM 145, 2013, 111 – 118.*

[7] J. Möck, J. Weiland. *Advancing Virtual Commissioning with Variant Handling. In: Proc. ASIM Meeting STS/GMMS 2014, Reutlingen, ARGESIM Report 42 / AM 149, 2014, 7 – 13.*

[8] *D. Pascheka. Implementierung eines graphischen SES-Editors mit integriertem Pruning Algorithmus in der MATLAB/Simulink Umgebung. (Implementation of a Graphical SES Editor with Integrated Pruning Algorithm within MATLAB/Simulink. In German) Bachelor Thesis, Hochschule Wismar – Univ. of Applied Sciences, RG CEA, 2014.*

[9] *T. Pawletta, D. Pascheka, A. Schmidt. System Entity Structure Ontology Toolbox for MATLAB/Simulink: Used for Variant Modelling. In: F. Breitenecker, I. Troch eds., Proc. of MATHMOD 2015 - 8th Vienna Int. Conf. on Mathematical Modelling, Vienna, Austria, 2015, 2 pages (submitted for publishing).*

[10] *Protégé. A free, open-source ontology editor and framework for building intelligent systems. [Online]* http://protege.stanford.edu/, (Accessed on: 01/04/2014).

[11] *A.*Schmidt, T. Pawletta. *Ein Ontologie-basierter Modellierungs- und Simulationsansatz am Beispiel der ressourceneffizienten Planung spanender Prozessketten. (An Ontology-Based Modeling and Simulation Approach using the Example of Planning Resource Efficiency Manufacturing Process Chains. In German) In: Proc. 15th ASIM Fachtagung Simulation in Produktion und Logistik, Paderborn, HNI-Verlagsschriftenreihe Bd. 316, 2013, 481 – 490.*

[12] A. Schmidt, U. Durak, C. Rasch, T. Pawletta. *Model-Based Testing Approach for MATLAB/Simulink using System Entity Structure and Experimental Frames.* In: Proc. of Spring Simulation Multi-Conf., Alexandria/VA, USA, April 12th-15th, 2015, 8 pages (submitted for publication).

[13] *T. Schwatinski, T. Pawletta, S. Pawletta. Flexible task oriented robot controls using the System Entity Structure and model base approach. SNE - Simulation Notes Europe, Vol. 22(2012)2, 107 – 114.*

[14] *T. Schwatinski, A. Schmidt, T. Pawletta.* Tiny SES Toolbox for MATLAB/Simulink. 2012, [Online] http://www.mb.hs-wimar.de/cea/SES_Tbx/.

[15] *T. Schwatinski, T. Pawletta. Ontologische Modellierung und Modellgenerierung in der MATLAB/Simulink Umgebung – Die Tiny SES Toolbox. (Ontological Modeling and Model Generation within MATLAB/Simulink – The Tiny SES Toolbox. In German) In: Proc. ASIM Meeting STS/GMMS, Düsseldorf, ARGESIM Report 41 / AM 145, 2013, 57 – 64.*

[16] H. Stuckenschmidt. *Ontologien (Ontologies. In German), 2nd Ed.* Springer Pub., 2011.

[17] The MathWorks. *Simulink User's Guide R2014a*, Natick, USA, 2014.

[18] M. Utting, B. Legeard. Practical Model-Based Testing: A Tools Approach. Morgan Kaufman Pub. Inc., 2007.

[19] B.P. Zeigler. *Multifaceted Modeling and Discrete Event Simulation,* Academic Press, 1984.

[20] B. P. Zeigler, H. Prähofer, T.G. Kim. *Theory of Modeling and Simulation 2nd Ed.* Academic Press, 2000.

[21] B.P. Zeigler, P. Hammonds. *Modeling and Simulation-Based Data Engineering.* Elsevier Academic Press, 2007.

[22] B.P. Zeigler, H.S. Sarjoughian. *Guide to Modeling and Simulation of Systems of Systems.* Springer Pub., 2013.