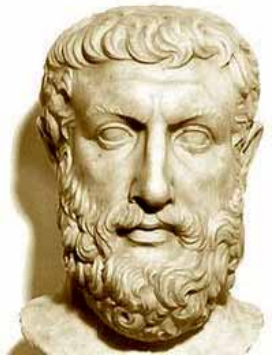
SNE SIMULATION NOTES EUROPE

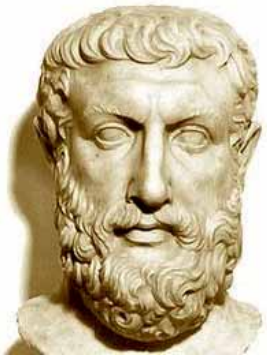
SNE Special Issue

Ontologies in Modelling and Simulation



Ontology is the philosophical study of the nature of being, becoming, existence, or reality, as well as the basic categories of being and their relations.

In computer science and information science, an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse.



Volume 24 No.2 Aug. 2014
doi: 10.11128/sne.24.2.1024

Print ISSN 2305-9974
Online ISSN 2306-0271

Journal on Developments and Trends in Modelling and Simulation

Membership Journal for Simulation Societies and Groups in EUROSIM

ARGESIM

SNE Editorial Board

SNE - Simulation Notes Europe is advised and supervised by an international scientific editorial board. This board is taking care on peer reviewing and handling of *Technical Notes*, *Education Notes*, *Short Notes*, *Software Notes*, *Overview Notes*, and of *Benchmark Notes* (definitions and solutions). At present, the board is increasing:

Author's Info

Authors are invited to submit contributions which have not been published and have not being considered for publication elsewhere to the **SNE** Editorial Office. Furthermore, SNE invites organizers of EUROSIM conferences to provide post-conference publication for the authors of their conference (with peer review).

SNE distinguishes different types of contributions (*Notes*):

- *Overview Note* – State-of-the-Art report in a specific area, up to 14 pages, only upon invitation
- *Technical Note* – scientific publication on specific topic in modelling and simulation, 6 – 8 (10) pages
- *Education Note* – modelling and simulation in / for education and e-learning; max. 6 pages
- *Short Note* – recent development on specific topic, max. 4 p.
- *Software Note* – specific implementation with scientific analysis, max 4 pages
- *Benchmark Note* – Solution to an ARGEIM Benchmark; basic solution 2 pages, extended and commented solution 4 pages, comparative solutions 4-8 pages

Further info and templates (doc, tex) at **SNE's** website.

SNE Contact & Info

→ *www.sne-journal.org*
office@sne-journal.org, *eic@sne-journal.og*
SNE Editorial Office, ARGESIM / dwh Simulation Services,
Neustiftgasse 57-59, 1070 Vienna, Austria

SNE Simulation Notes Europe
ISSN SNE Print ISSN 2305-9974, SNE Online ISSN 2306-0271
WEB: → *www.sne-journal.org*, DOI prefix 10.11128/sne
Scope: Technical Notes, Short Notes and Overview Notes on developments and trends in modelling and simulation in various areas and in application and theory; benchmarks and benchmark documentations of ARGESIM Benchmarks on modelling approaches and simulation implementations; modelling and simulation in and for education, simulation-based e-learning; society information and membership information for EUROSIM members (Federation of European Simulation Societies and Groups).
Editor-in-Chief: Felix Breitenecker, Vienna Univ. of Technology, Inst. f. Analysis and Scientific Computing, Div., Math. Modelling and Simulation, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; *Felix.Breitenecker@tuwien.ac.at*, *eic@sne-journal.org*
Layout / Administration: J. Tanzler, F. Preyser, T. Wobruba; C. Wytrzens, R. Leskovar et al.; Math. Modelling and Simulation Group, Vienna Univ. of Technology, Wiedner Haupstrstasse 8-10, 1040 Vienna , *office@sne-journal.org*
Print SNE: Grafisches Zentrum, TU Vienna, Wiedner Hauptstrasse 8-10, 1040, Vienna, Austria
Online SNE: ARGESIM /ASIM, c.o. dwh Simulation Services, Neustiftgasse 57-59, 1070 Vienna, Austria
Publisher: ARGESIM Arbeitsgemeinschaft Simulation News - Working Committee Simulation News, Neustiftgasse 57-59, 1070 Vienna, Austria; → *www.argesim.org*, *info@argesim.org* on behalf of ASIM(→ *www.asim-gi.org* and EUROSIM → *www.eurosim.info*

Editorial

Dear Readers – *Traditionally the second issue of each SNE Volume is a Special Issue – also in 2014 SNE continues this tradition, with a special issue on Ontologies in Modeling and Simulation. Additionally, this issue continues the new submission strategy which invites individual submissions and post-conference publications from EUROSIM societies' conferences. The origin of the contributions in this special issue SNE 24(2) are publications from the special track 'Ontologies in Modeling and Simulation' presented at ASIM SST 2014 Symposium Simulation Technique 2014 in Berlin – for details see Special Issue Editorial of the guest editors (ASIM German Simulation Society).*

I would like to thank all authors for their contributions, and the ARGESIM SNE staff for helping to manage the SNE administration and the improved SNE layout and extended templates for submissions (now also tex). Special thanks to the guest editors of this special issue, to Thorsten Pawletta from Wismar University of Applied Sciences, and to Umut Durak, Deutsches Zentrum für Luft und Raumfahrt e.V. (DLR), Braunschweig, for compiling this very interesting issue bridging modelling and simulation at the one side, and computer science at the other side.

Felix Breitenecker, SNE Editor-in-Chief, eic@sne-journal.org; felix.breitenecker@tuwien.ac.at

Contents SNE 24(2)

Special Issue

'Ontologies in Modeling and Simulation'

SNE doi: 10.11128/sne.24.2.1024

Editorial SNE Special Issue 'Ontologies in Modeling and Simulation'

In the last 10 years, the advances in semantic web have influenced modeling and simulation (M&S). Gruber's [1] definition that *"ontology is a formal specification of conceptualization"* has been well accepted by the M&S community. In 2004, Miller, et al. first introduced how ontologies could be used for M&S [2].

Since then, various M&S researchers have investigated methodologies that employ ontologies. Some of the early M&S ontologies to be mentioned include *Discrete Event Modeling Ontology* [3], as a general modeling ontology, and *Trajectory Simulation Ontology* [4], which targets a particular domain. This Special Issue (SI) of SNE, on the other hand, aims at presenting some of the recent developments and applications of M&S ontologies. Five of the six papers that constitute this SI are revised scientific and technical papers that were presented at the 'Ontologies in Modeling and Simulation' special track of the ASIM 2014 – 22nd Symposium Simulationstechnik which was held in Berlin, 3-5 September, 2014.

The *first* paper presents an ontology-based system modelling approach for MATLAB/Simulink. Model structures are defined using the System Entity Structure (SES) ontology, originally introduced by Zeigler, and references to basic models that reside in a repository. In addition to the ontology and the developed modeling environment, methods for an automatic generation of simulation models using the ontology and basic blocks or models, which are defined within MATLAB/ Simulink, are discussed.

The *second* paper introduces the utilization of ontologies for objective flight simulator fidelity evaluation, which fundamentally addresses the simulation fidelity problem. The evaluation is based on the comparison of a simulator and actual flight through quantitative measures and conducted via testing. As such, Durak, Schmidt and Pawletta employ ontologies as metamodels for introducing *Model Based Testing* practices for fidelity evaluation.

The *third* paper is intended to support the design and management processes of seaport container terminals with a simulation ontology. Lange, et al. aim for a quicker and more flexible simulation model building process that requires less know-how of specific simulation software. Therein, they propose an ontology-based model generation, which is enabled by a framework consisting of a user interface, a library for atomic models and a specific model generation method.

In the *fourth* paper, ontologies are utilized for capturing the knowledge of different car concepts, which is scattered across various sources, in a domain model. The domain model links various forms of information, objects, and properties, etc. and is the basis for supporting overall vehicle simulation studies. Krausz, Zimmer and Reuss present an implementation of such an approach, called OverNight Testing (ONT). With the underlying domain model, ONT provides capabilities to specify overall vehicle configurations out of a huge number of vehicle concepts and to test and assess these configurations by simulation studies.

Hocaoglu, in the *fifth* paper, introduces an ontology-based modeling approach in which entities have spatial and temporal dimensions. The modeling approach is augmented with a reasoning mechanism, which allows the managing of entity behaviors relying on reasoning results. The ontology-based modeling approach is supported by an agent-driven simulation language for high level action descriptions, higher order world envisionment, dynamic relation management and reasoning.

Finally, in the *sixth* paper, Popper, et al. present a formal modeling approach, inspired by a project known as *Balanced Manufacturing (BAMA)*, with the aim of monitoring, predicting and optimizing energy and resource demands. Ontology is not employed in the strict sense, but rather a formal approach to modularize complex production systems is introduced. System components are segmented in so-called *cubes*. The basic concept of cubes, their latitude and limitations are discussed.

References

[1] T. Gruber. *A Translation Approach to Portable Ontology Specification*. Knowledge Acquisition, Bd. 5, No. 2, 1993, 199-220.

[2] P. Fishwick, G. Maramidze, A. Sheth, J. Miller. *Investigating Ontologies for Simulation Modeling*. In: Proceedings of 37th Annual Simulation Symposium, Arlington, VA, USA, 18-22 April, 2004, 55 - 63.

[3] J. Miller, G. Baramidze. *Simulation and the Semantic Web*. In: Proceedings of the 2005 Winter Simulation Conference, Orlando, FL, USA, 2005, 2371 - 2377.

[4] U. Durak, H. Oguztuzun, S. Ider. *An Ontology for Trajectory Simulation*. In: Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, 2006, 1160 - 1167.

Thorsten Pawletta, *thorsten.pawletta@hs-wismar.de*
Research Group of Computational Engineering and Automation (CEA), Wismar University of Applied Sciences, Philipp-Müller-Straße 14, D-23966 Wismar, Germany

Umut Durak
Inst.of Flight Systems, Deutsches Zentrum f. Luft und Raumfahrt e.V. (DLR), Lilienthalpl. 7, 38108 Braunschweig, Germany

SNE Reader's Info

Simulation Notes Europe publishes peer reviewed *Technical Notes*, *Short Notes* and *Overview Notes* on developments and trends in modelling and simulation in various areas and in application and theory, with main topics being simulation aspects and interdisciplinarity.

Individual submissof scientific papers are welcome, as well as post-conference publications of contribtions from conferences of **EUROSIM** societies.
Furthermore **SNE** documents the **ARGESIM Benchmarks** on *Modelling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions (*Benchmark Notes*). Special *Educational Notes* present the use of modelling and simulation in and for education and for e-learning. **SNE** is the official membership journal of **EUROSIM**, the Federation of European Simulation Societies. A News Section in **SNE** provides information for **EUROSIM** Simulation Societies and Simulation Groups.

SNE is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With **Online SNE** the publisher **ARGESIM** follows the **Open Access** strategy, allowing download of published contributions for free. Since 2012 **Online SNE** contributions are identified by a DOI (Digital Object Identifier) assigned to the publisher **ARGESIM** (DOI prefix 10.11128). **Print SNE**, high-resolution **Online SNE**, full **SNE Archive**, and source codes of the *Benchmark Notes* are available for members of **EUROSIM** societies.

SNE Print ISSN 2305-9974, SNE Online ISSN 2306-0271
SNE Issue 24(2) August 2014 doi: 10.11128/sne.24.2.1024
→ *www.sne-journal.org*
office@sne-journal.org, eic@sne-journal
SNE Editorial Office, c/o ARGESIM / DWH, Neustiftgasse 57-59, 1070 Vienna, Austria

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.

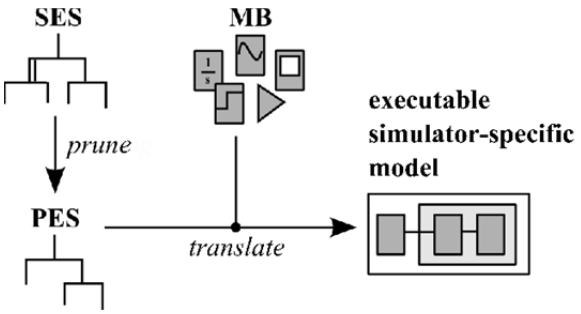


Figure 1: SES/MB framework according to [20].

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.

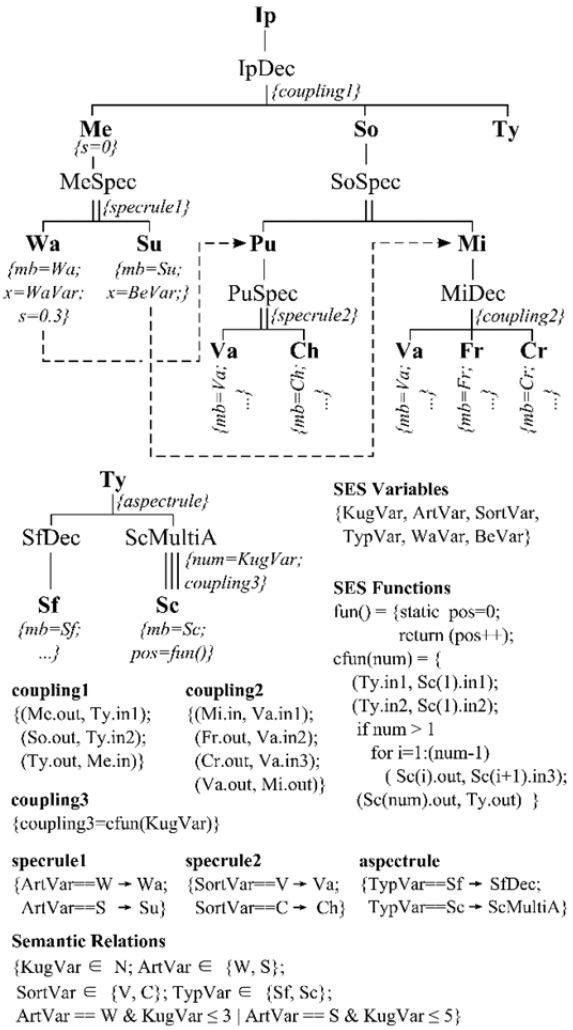


Figure 3: mSES for the ice cream portion example.

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.

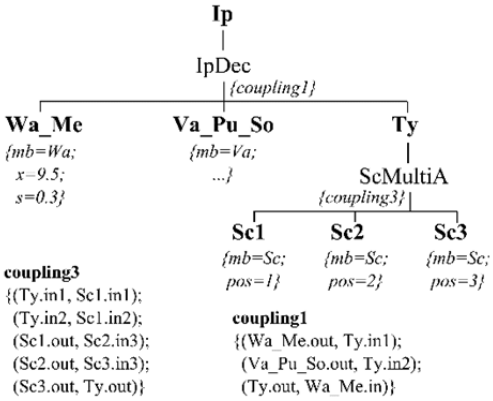


Figure 4: A unique, valid PES of the mSES in Figure 3.

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.

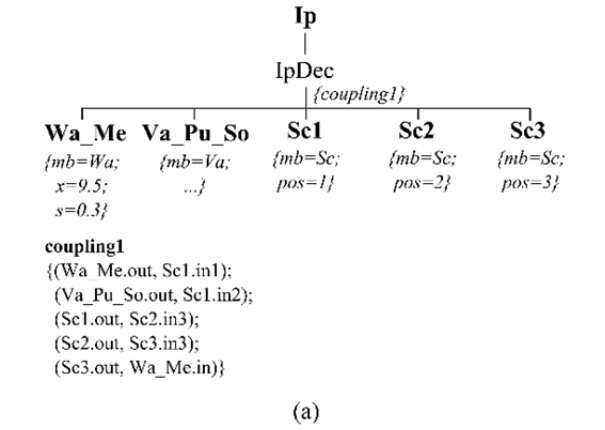


Figure 5: (a) FPES of PES in Fig. 4; (b) corresponding system structure.

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.

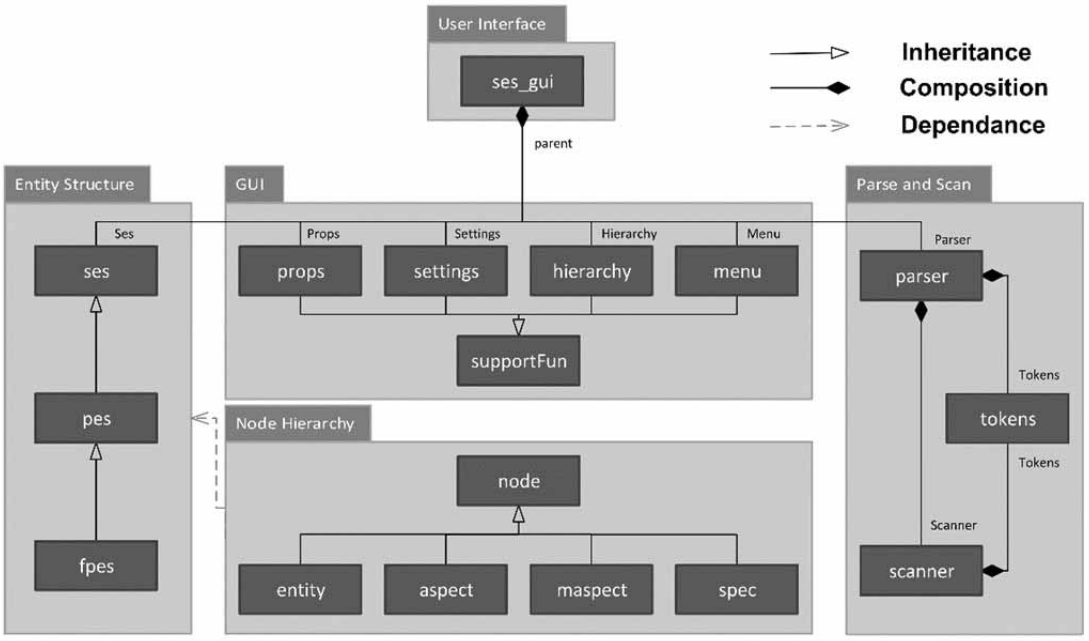


Figure 6. Class structure of SES toolbox.

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.

Ontology for Objective Flight Simulator Fidelity Evaluation

Umut Durak[1*], Artur Schmidt[2], Thorsten Pawletta[2]

[1] German Aerospace Center (DLR), Institute of Flight Systems, Lilienthalplatz 7, 38100 Braunschweig, Germany; *umut.durak@dlr.de

[2] University of Wismar, FIW / MVU, FG CEA, PF 1210, 23952, Wismar, Germany

Simulation Notes Europe SNE 24(2), 2014, 69 - 78
DOI: 10.11128/sne.24.tn.10242
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.);
Revised: Oct. 20, 2014; Accepted: October 30, 2014;

Abstract. The term simulator fidelity has become enormously important in the scope of simulation research, when assessing training efficiency and the transfer of training to real flight. It is defined as the degree to which a flight simulator matches the characteristics of the real aircraft. Objective simulator fidelity provides an engineering standard, by attacking the fidelity problem with comparison of simulator and the actual flight over some quantitative measures. Research flight simulators encompass some differences from commercial flight simulators. They require high flexibility and versatility concerning the cockpit layout and visual and motion systems, as well as flight simulation models. It should be easy to modify the flight simulation model or other software and hardware components of the simulator. To support this, there is a need for a flexible automated test methodology, in order to determine the fidelity of the most relevant simulator subsystems, since they are often modified during the life cycle of the simulator. This methodology not only shall support automated execution but also enable automated generation of the test cases which are subject to change as well as simulator components. The Institute of Flight Systems (FT) at the German Aerospace Center (DLR) has a reconfigurable flight simulator, the Air Vehicle Simulator (AVES), for research of rotorcraft and fixed-wing aircraft.
The study reported in this paper adopts a Model Based Testing approach to tackle the high flexibility requirement of AVES. The outcome of the paper is a metamodel for model-based objective flight simulator evaluation. Metamodeling has been carried out in two levels. An Experimental Frame Ontology (EFO) has been developed adopting experimental frames from Discrete Event System Specification (DEVS), and as an upper ontology to specify a formal structure for a simulation test. Then in Objective Fidelity Evaluation Ontology (OFEO) that builds upon EFO, domain specific meta-test definitions are captured.

Introduction

Since the late 1920s, when Edward Link built the 'Blue Box' [1], flight simulators have been important elements of aviation. Flight simulators became well accepted as training aids by many aircraft operators before the digital era. Highly sophisticated flight simulators have been employed commercially within civil and military flight training organizations in order to enhance pilot skills.

In the 1980s, the aeronautics research community started using flight simulators for developing and experimenting advanced concepts and conducting aviation human factors research. Some of the first examples of research flight simulators include ATTAS Ground-Based Simulator from German Aerospace Center (DLR) [2] [3], National Aerospace Agency (NASA) Crew Vehicle Systems Research Facility in Ames Research Center [4] and Visual Motion Simulation and Cockpit Motion Facility at the Langley Research Center [5]. Some more recent examples are the Air Vehicle Simulator (AVES) of DLR [6], HELIFLIGHT at the University of Liverpool [7], NASA Ames Vertical Motion Simulator (VMS) [8] and International Research Institute for Simulation, Motion and Navigation (SIMONA) of Delft University of Technology [9].

The authors define fidelity in flight simulation as the degree to which a flight simulator matches the characteristics of the real aircraft. As its effect on training efficiency and transfer of training to real flight became better understood, fidelity became a more important research subject [10]. Objective simulator fidelity assessment provides an engineering standard to qualify the degree of fidelity through objective measures. It approaches the fidelity problem with comparison of simulator and the actual flight over some quantitative cues.

Requirements for research flight simulators encompass some differences from commercial flight simulators.

They require high flexibility and versatility concerning the cockpit layout and visual and motion systems, as well as flight simulation models. They must allow easy modification of the flight simulation model or other software and hardware components of the simulator. In order to efficiently determine the fidelity of subsystems that are often modified during the life cycle of the simulator, there is a need for a flexible automated test methodology.

This methodology is required to automate not only the execution, but also the test case generation. While there are standard sets of test cases for objective flight simulator evaluation, each modification of simulator components asks for either a different subset of a standard test set or modifications in standard test specifications. Therefore, test cases are also required to be easily modifiable, as well as the components of a research simulator.

Automated testing can be applied through the use of software to control the execution of tests and a comparison of actual outcomes to the predicted ones. Available test data taken from aircraft are used as input signals to the simulator and the output signals of the simulator are compared to the measurements to be presented for the evaluator in a smart format. Braun and Galloway [11] reported their automated fidelity test system that compares directly the flight test results and manual execution of flight tests in simulators.

Wang et al. [12] [13] presented *Automated Test System* (ATS) that measure force function, evaluation function and transport delay with its non-intrusive interface with operator station. Jarvis et al. [14] summarizes the efforts on validation of sensory cues, motion cues, vibration and sound cues, visual cues, transport delays and flight dynamics models in flight simulators.

Previous efforts regarding automated testing for objective flight simulator evaluation utilized fixed test descriptions. The presented automated testing infrastructures contributed flawless execution of the tests. But they did not attack automation of test case generation. The bridge between the state of the art Model Based Testing (MBT) practices and automated flight simulator testing is still missing. MBT can be introduced as the idea of automating test case generation from a test model rather than implementing test cases manually [15].

Thus, the test case generation is made more flexible. Metamodeling is employed to capture the domain specific concepts and constraints for building test models.

Then test modeling is used to specify test cases, and these test models are translated automatically to executable test cases [16]. DLR intends to adopt an MBT approach in flight simulator domain and hereby provide a methodology for flexible automated test case generation. Therefore a metamodel is required for objective flight simulator fidelity evaluation.

A metamodel is defined as an explicit model of constructs and rules that are used to define a model [17]. Following Gruber [18], definition of ontology is "explicit specification of shared conceptualization". Moreover, metamodels are categorized as ontologies that are used by modelers [17].

Here, the test case can be defined as a sequence of input stimuli that will be fed to the System Under Test (SUT), namely test inputs and the expected behavior of the system, namely test oracle (**Figure 1**) [19].

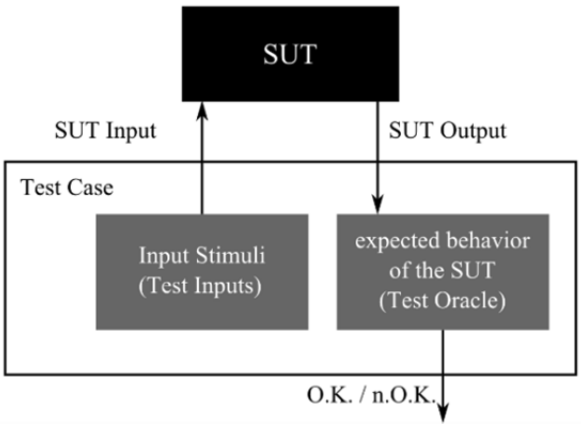


Figure 1. Test Case Structure.

Moser et al. [20] stressed that ontologies as machine-readable domain knowledge, which can be utilized for test case generation. Then Nguyen et al. [21] presented a framework for ontology driven test case generation in the context of multi-agent systems. Adopting these ideas, ontologies are employed to structure meta-test definitions.

The domain knowledge about the objective validation of simulator systems including the rules for assessing the results of test runs is captured in ontologies.

Zeigler and his colleagues developed the concept of *Experimental Frame* (EF) [22] [23]. An EF defines the conditions under which a model is to be examined. It comprises of an input generator, a verifier for the desired conditions and an analyzer for the outputs.

Following Zeigler et al. [23], the EF is critical for evaluating the model validity. Traoure and Muzy in [24] and Foures et al. in [25] published the usage of the EF approach for specifying invariant validation experiments.

In this research, metamodeling has been carried out on two levels. An *Experimental Frame Ontology* (EFO) has been developed as an upper ontology to specify a formal structure for generic simulation test model. Then in *Objective Fidelity Evaluation Ontology* (OFEO) that builds upon EFO, domain specific meta test definitions are captured. Protégé [26] is used as the ontology development environment and ontologies are developed using *Ontology Web Language* (OWL).

This paper will present these ontologies after introducing a background on objective fidelity evaluation, experimental frames and ontologies in general.

In this paper, first a background will be introduced on objective fidelity evaluation, EF and ontologies. Then EFO and OFEO will be presented. The paper will end with concluding remarks.

1 Background

1.1 Objective fidelity evaluation

Fidelity is regarded as a multivariate construct with no consensus among researchers on a single index of measurement or definition and it is strongly related to the training task to be performed with the simulator.

There are two approaches to measure simulator fidelity; the subjective and objective approaches [12]. The subjective approach tries to identify the degree of realism felt by the user. User feedback is usually collected using subjective rating scales [27].

Although subjective scales are valuable, it is hard to generalize across scales because of the individual opinions and bias of those providing assessments [12]. Objective approaches attack the fidelity problem with of simulator and the actual flight over some quantitative cues.

'ICAO 9625 Manual of Criteria for the Qualification of Flight Training Devices' [28] is the well accepted global standard for qualification of flight training devices. The standard specifies seven types of fidelity that correspond to a capability level to provide a certain type of training.

For example, simulators classed as 'Type 1' can be used for all training tasks used during completion of Private Pilot License (PPL) training, whereas 'Type 7' is required for some of the training tasks used when awarding 'Type Rating'. Appendix B of the standard specifies the test cases for objective validation of simulators. These test cases include comparison of results from tests conducted in the simulator and aircraft validation data.

The Royal Aeronautical Society (RAeS) published 'Aeroplane Simulation Training Device Evaluation Handbook Vol. 1 Objective Testing' [29] to ease the implementation and enhance the understanding of objective tests introduced in ICAO 9625. It provides further discussions about the implementation of each test and introduces some example cases with some plots. ICAO 9625 provides tables that specify each test case with parameters, tolerances and flight conditions. Table 1 shows an example test specification from the standard, for testing the minimum radius.

Test	Tolerance	Type						
		1	2	3	4	5	6	7
Minimum radius turn	±0,9m (3ft) or ±20% of aeroplane turn radius					✓		✓

Table 1: Sample Test Specification from ICAO 9625 [28]

This effort takes ICAO 9625 as a baseline to define test cases as they present a shared understanding of experts in the field. Tests are grouped under performance, handling qualities, motion system, visual system and sound system. Among these tests, those regarding performance and handling qualities are related to flight dynamics models, and have no other subsystem or device dependencies. For this reason, they are considered to better suit automation. Therefore, as a first step, the current research addresses these groups.

The RAeS introduces the benefits of employing automatic testing in objective fidelity evaluation as repeatability, ease and rapidity of conducting tests. The RAeS handbook [29] specifies the features of an automatic testing system as initializing the simulator with the test initial conditions, trimming the aircraft, creating the stimulus if required, using flight controls and finally checking the simulator output against test criteria.

1.2 Experimental frame approach

The EF approach was originally introduced by Zeigler in [22] in context with the *Discrete Event System Specification* (DEVS). The objective is the explicit separation between the model and the experiment. Moreover, an EF specifies a limited set of circumstances under which a model is to be observed. Currently, the EF approach belongs to the state of the art and it is used in many modelling and simulation projects including validation experiments [24] [25] [30] [31]. Following Zeigler [22], the formal specification of the EF is given by the 7-tuple:

$$EF = < T, I, O, C, \Omega i, \Omega c, SU >$$

where:

T is the time base
I is the set of input variables
O is the set of output variables
C is the set of control variables
Ωi is the set of admissible input segments
Ωc is the set of admissible control segment
SU is a set of summary mappings

The EF can be implemented in various ways. Zeigler [22] recommends implementing the EF as a coupled system consisting of a generator, acceptor and a transducer that is connected to a SUT. In our context, the SUT is always a model. For this reason, it is called *Model Under Test* (MUT). Figure 2 illustrates such a realization of EF coupled to a MUT schematically.

Test inputs are produced by a generator. The set of admissible input segments influences MUT's behavior. The acceptor and transducer form the test oracle. Based on output variables, the transducer calculates outcome measures in the form of performance indices, comparative values, statistics etc.

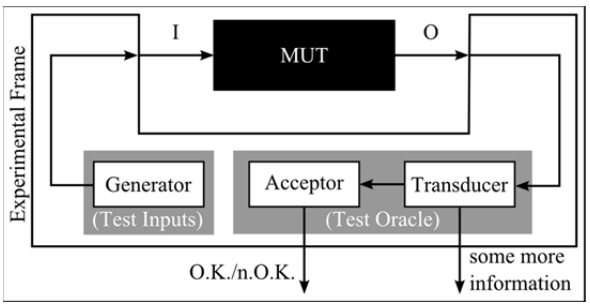


Figure 2: Illustration of EF with MUT.

The acceptor corresponds to a decision unit that decides if an experiment is valid or not. For this purpose, the acceptor monitors its inputs and maps them to a specified admissible control segment. In case of violation of the admissible control segment the experiment will not be accepted. Beside control variables, the input of an acceptor can be output variables or outcome measures.

The EF approach defines a uniform structure for a systematic experiment specification. The specification has to be coded in the description of an EF. This means that each kind of experiment needs the definition of a distinct EF.

1.3 Ontologies

Knowledge in a domain is formalized using concepts, relations, functions, axioms and instances in an ontology. Concepts can be anything about which something is said, and therefore, can be a description of a task, function, action, strategy etc. Taxonomies are widely used to organize the ontological knowledge in domain using generalization/specialization relationship through simple/multiple inheritance.

Relationships represent a type of interaction between the concepts of the domain and functions can be regarded as a special kind of relation. Axioms on the other hand are used to model sentences that are always true. They are added to ontology for several purposes, such as constraining the information contained in the ontology, verifying its correctness or deducting new information. Instances are the terms that are used to represent the elements of the domain. They actually represent the elements of the concepts [32].

Ontologies in engineering domain have been developed for various purposes including specifying engineering information systems, integration of engineering applications, supporting engineering design and development. The first efforts on developing engineering ontologies were in the 1990's. The 'PhysSys' [33] was one of the first engineering ontologies based upon system dynamics theory that is practiced in engineering modeling, simulation and design. The PhysSys was developed to formally define how design engineers or the end users of *Computer Aided Engineering* (CAE) systems understand their domain and to provide a foundation for the conceptual schema for data structuring in engineering databases, libraries and other CAE information systems [33] [34].

The ideas formalized in PhysSys provided a base for the development of a library of reusable models for engineering and design.

Fishwick and Miller in [35] discussed the venues of ontology use in modeling and simulation. One of the late examples of ontology use in modeling and simulation is reported by Durak et al. [36] [37]. The group enabled simulation reuse over an ontology driven methodology.

Another ontology-based modeling and simulation approach was established by Zeigler with the System Entity Structure and Model Base (SES/MB) framework [22] [23] [38] [39].

Today the SES is an ontology framework for conceptual system modeling and for specification of a set of modular hierarchical system structures and parameter settings.

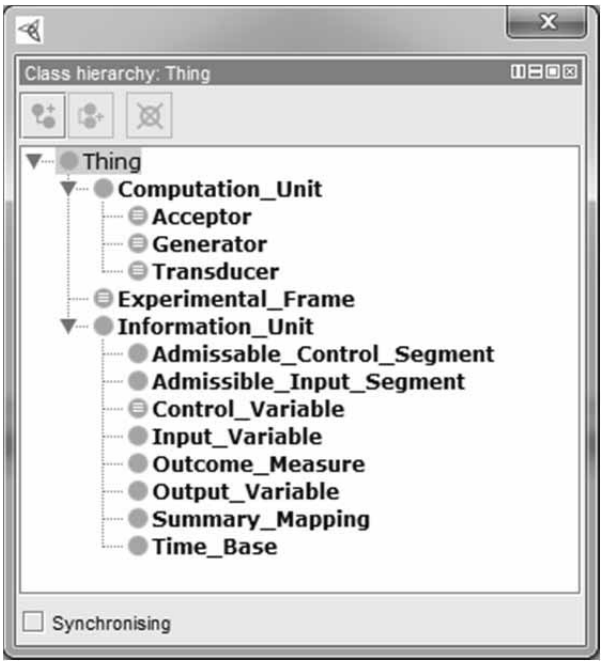


Figure 3: Entity Hierarchy of the Experimental Frame Ontology.

2 Experimental Frame Ontology

The EFO forms the upper level of the metamodel for objective flight simulation evaluation. The previously introduced EF approach is used to specify a formal structure of generic test cases. Hence, every test case has to be specified according to the EF definition in the Section 1.2.

Figure 3 illustrates the entity hierarchy of the EFO in Protégé. The first layer consists of three entities: Computational Unit, Informational Unit and the EF. Computational Units comprises the generic Acceptor, Transducer and Generator which will be presented as executable blocks in a test case. The Information Unit defines basic entities of an EF. The Experimental Frame entity thus conforms to the actual EF.

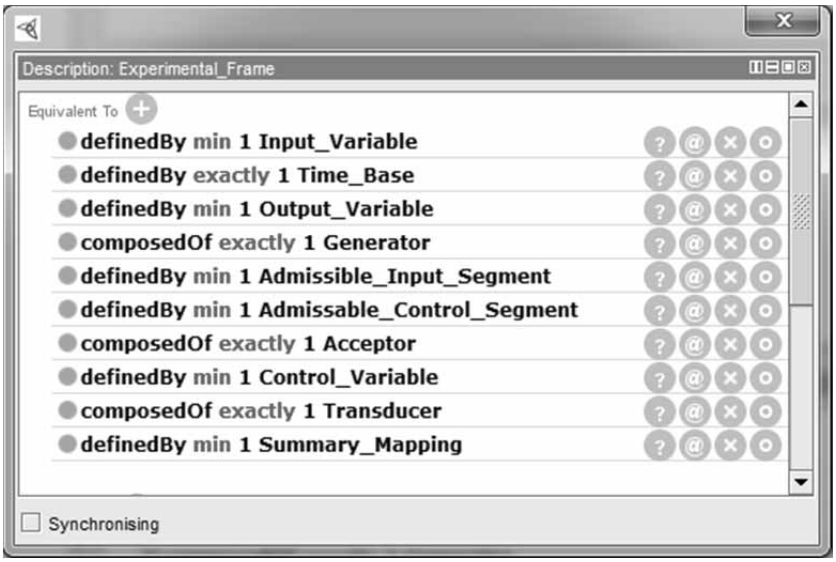


Figure 4: Description of a Generic Experimental Frame.

Furthermore particular properties are implemented to define the relations between the entities. For example the properties *composedOf* and *definedBy* makes clear that any EF is a composition of Computational Unit and is defined by the Informational Units.

As a result we obtain a generic EF which conforms to a generic test case. Thus, any test case will have the unique structure as shown in Figure 4 on its top level. The EFO forms the basis for the OFEO that will define test cases in detail.

3 Objective Fidelity Evaluation Ontology

OFEO is constructed by extending the upper level EFO that specifies any test case that will be applied to MUT using experimental frames formalism. The hierarchy of OFEO using Protégé is depicted in Figure 5. The elements from EFO can be traced in this hierarchy.

Each objective validation test case described in ICAO 9625 under performance and handling qualities are specified by an experimental frame. Thus, each test possesses a Generator, Transducer and an Acceptor. The specification of these three entities will inherently describe how this specific test will be exercised. These three entities will constitute the automatic test system.

Following the features of automated test systems introduced in the RAeS Handbook [29], the Generator is described as the component to initialize the test with initial conditions and trim the aircraft and create the stimulus following the ones from the flight test using the flight controls.

Hence, the Generator is interpreted as test independent. On the other hand, the Transducer is described as the component that will compute Outcome Measures that are required for the Acceptor for a specific test.

As an example, the Minimum Turn Radius test requires a Simulated Turn Radius to be computed from a simulation output. Or likewise, Rate of Turn versus Nosewheel Steering Angle test requires Simulated Turn Rate value to be computed.

So, a specific Transducer is defined for every test. Lastly, the Acceptor is described as the component that checking the MUT against test criteria. Since every test has a particular criterion, an Acceptor is defined for each test. Accordingly, we are expecting to have particular Control Variables for each test.

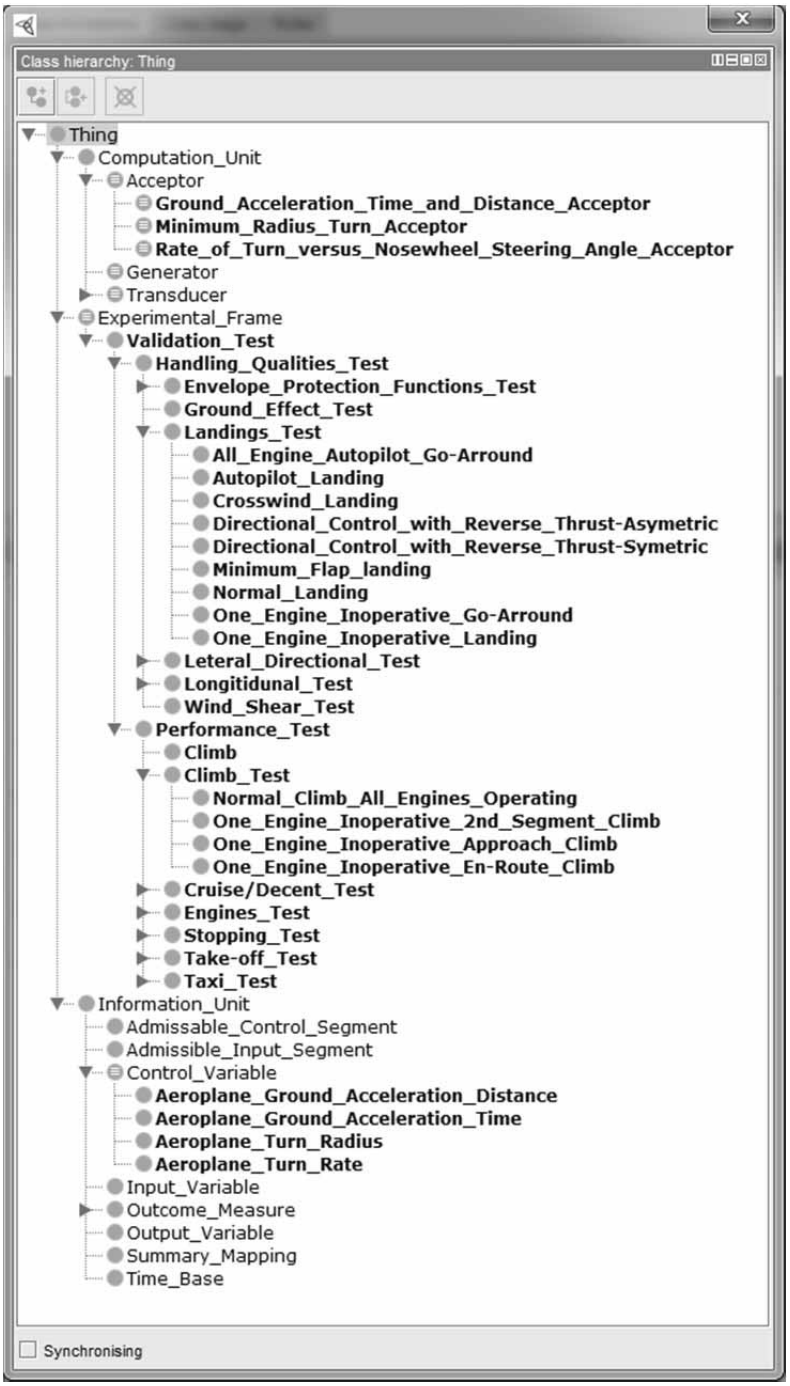


Figure 5: Objective Fidelity Evaluation Ontology Hierarchy.

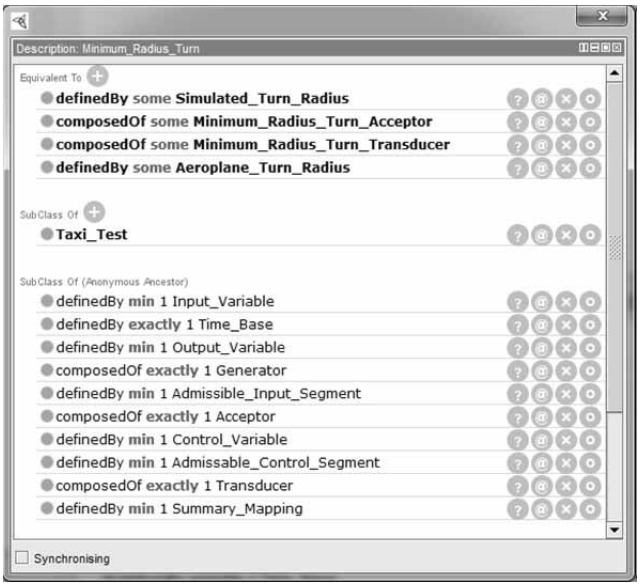


Figure 6: Minimum Radius Turn Test Description.

Figure 6 presents an example test description in Protégé. The Minimum Turn Radius Test is specified with a specific Acceptor, Transducer and Control Variables, namely Simulated Turn Radius and Aeroplane Turn Radius. On the other hand, it inherits the properties of an experimental frame. So it will also have a Generator, Input Variables, Output Variables, Admissible Input Segments, Admissible Control Segments and a Summary Mapping. It is clear that input and output variables of the flight simulator are application specific but does not vary with test cases, so generic definitions are kept for these variables and admissible segments.

Minimum Radius Turn Transducer (Figure 7) is defined with an output Simulated Turn Radius while it also inherits the properties of a Transducer. It will be using Output Variables for computing the outcome measure. Since the computation of the outcome measure is largely implementation specific, ontology does not have any knowledge about it.

As an example, the Minimum Radius Turn Acceptor is depicted in Figure 8. Since each of the tests has distinct criteria, the Acceptors will have particular inputs. Accordingly, Minimum Radius Turn Acceptor is described with Simulated Turn Radius and Aeroplane Turn Radius inputs.

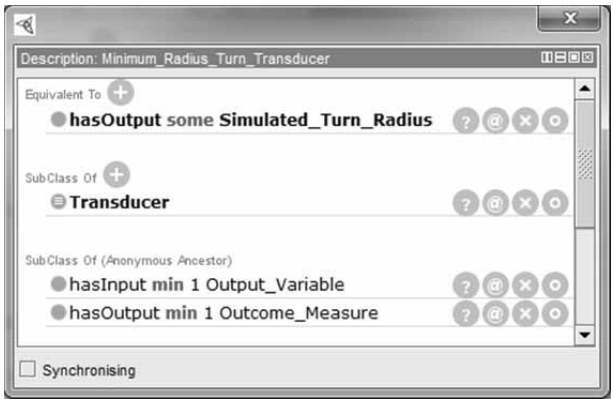


Figure 7: Minimum Radius Turn Transducer Description.

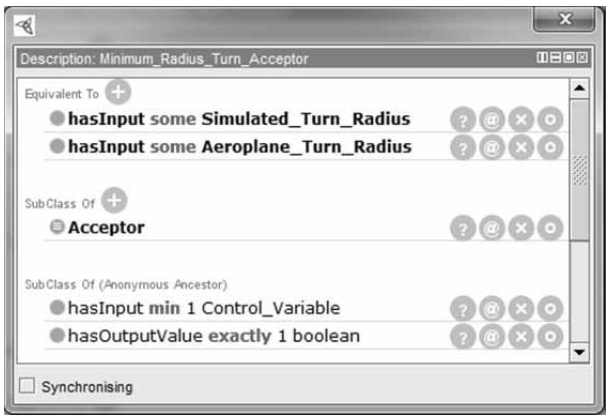


Figure 8: Minimum Radius Turn Acceptor Description.

On the other hand the output of the Acceptor is always a Boolean. It reports if the criterion is matched or not.

Semantic Web Rule Language (SWRL) [40] is used to formalize the acceptance criteria. SWRL can be regarded as an extension to OWL to specify rules for enhancing expressivity.

Thus rule-based reasoning over the knowledge captured in an ontology is possible. In this study, rules specify how the inputs of the Acceptor are used to compute if the test is successful or not. In

Figure **9**, the rule in the front windows says that Minimum Radius Turn Acceptor has a true output when the difference between the simulated and the real minimum turn radius is smaller than 20 %.

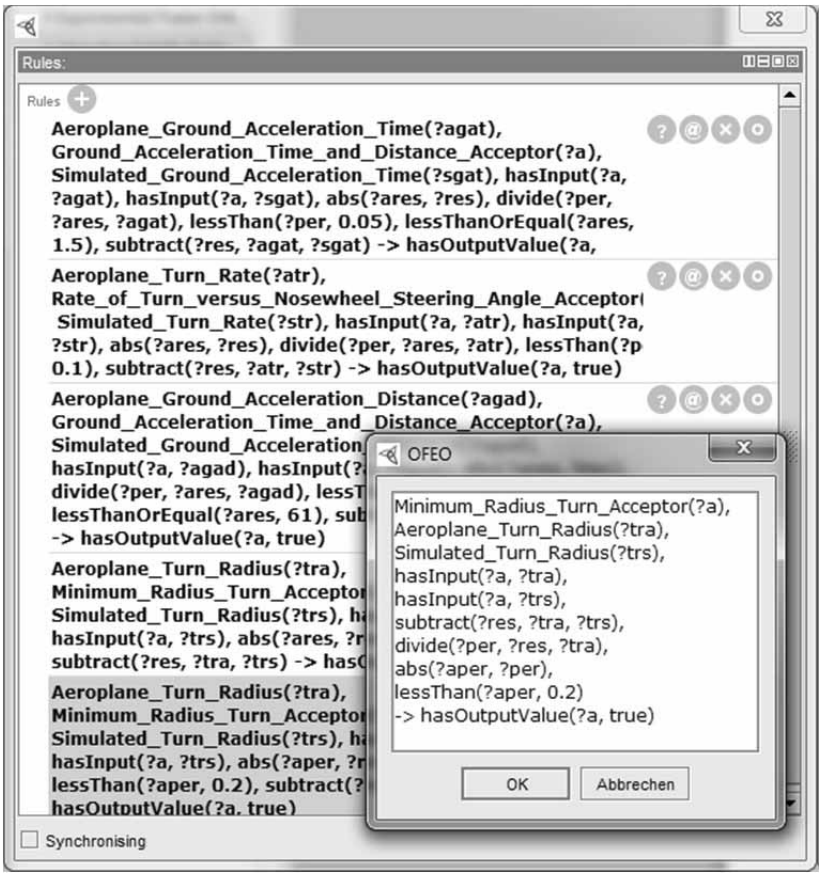


Figure 9: Rules for Acceptors.

While Web Ontology Language is used as the ontology language; Semantic Web Rule Language is employed to capture the rules. Protégé is utilized as the ontology development environment.

This effort assembled the semantic infrastructure for developing model based automated test methodology for simulator fidelity evaluation. The next step is to construct the toolset for developing the test models utilizing the presented metamodels. This toolset set shall also support model transformations to generate executable test cases and execution of these test cases.

Although Web Ontology Language, Semantic Web Rule Language are employed in this metamodeling step, the representation form of the knowledge captured in ontologies may vary in toolset implementation due to practical reasons like platform compatibility.

4 Conclusion

Research simulators require flexible and adoptable test methodologies to accommodate frequent changes to their components. This paper presents an ontology based metamodeling approach for adopting a Model Based Testing methodology for objective flight simulator evaluation.

Experimental Frames Ontology adopts the concept of Experimental Frames from Discrete Event Systems Specification, as an upper ontology to specify a formal structure for test cases.

Thus with Experimental Frames, concepts of Model Based Testing could be formally specified. This established a solid base for modeling specific test cases. Then in Objective Fidelity Evaluation Ontology that builds upon Experimental Frames Ontology, domain specific meta-test definitions are modeled.

References

[1] D. Allerton. *Principles of Flight Simulation.* John Wiley & Sons, Ltd, West Sussex, United Kingdom, 2009.

[2] P. Saager. *Real-Time Hardware-in-the-Loop Simulation for 'ATTAS' and 'ATTHeS' Advanced Technology Flight Test Vehicles*. in AGARD Guidance and Control Panel, 50th Symposium, Izmir, Turkey, 1990.

[3] S. Klaes. *ATTAS Ground Based System Simulator - An Update-*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Denver, CO, 2000.

[4] B. Sullivan and P. Soukup. *The NASA 747-400 Flight Simulator: A Natonal Reseource fir Aviation Safety Research*. In: AIAA Flight Simulation Technologies Conference, San Diego, CA, 1996.

[5] R. Smith. *A Description of the Cockpit Motion Facility and the Research Flight Deck Simulator*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Denver, CO, 2000.

[6] H. Duda, T. Gerlach, S. Advani and M. Potter. *Design of the DLR AVES Research Flight Simulator*. In: AIAA Modeling and Simulation Technologies (MS) Conference, Boston, MA, 2013.

[7] M. White and G. Padfield. *The Use of Flight Simulation for Research and Teaching in Acedemia*. In: AIAA Atmospheric Flight Mechanics Conference and Exhibit, Keystone, CO, 2006.

[8] S. Advani, D. Giovannetti and M. Blum. *Design of a Hexapod Motion Cueing System for NASA Ames Vertical Motion Simulator*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Monterey, California, 2002.

[9] O. Stroosma, R. van Paassen and M. Mulder. *Using the Simona Research Simulator for Human-Machine Interaction Research*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Austin, Texas, 2003.

[10] T. Longride, J. Bürki-Cohen, T. Go and A. Kendra. *Simulator Fidelity Considerations for Training and Evaluation of Today's Airline Pilots.* In: Proceedings of the 11th International Symposium on Aviation Psychology, Columbus, OH, 2001.

[11] D. Braun and R. Galloway, *Universal Automated Flight Simulator Fidelity Test System*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Rhode Island, 2004.

[12] C. Wang, J. He, G. Li and J. Han. *An Automated Test System for Flight Simulator Fidelity Evaluation*. Journal of Computers, vol. 4(11), 2009.

[13] C. Wang, J. Han, G. Li and H. Jiang. *Flight Simulator Fidelity Evaluation Automated Test System Analysis*. In: 2008 International Workshop on Education Technology and Training, Shanghai, China, 2008.

[14] P. Jarvis, D. Spira and B. Lalonde. *Flight Simulator Modeling and Validation Approaches and Pilot-in-the-loop Fidelity*. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Honolulu, Hawaii, 2008.

[15] J. Zander, I. Schieferdecker and P. Mosterman, A *Taxononomy of Model-Based Testing for Embedded Systems from Multipke Industry Domains.* In: Model-Based Testing for Embedded Systems, Boca Rato, CRC Press, 2012, pp. 3-23.

[16] A. Guduvan, H. Waselynck, V. Wiels, G. Durrieu, Y. Fusero and M. Schieber. *A Meta-Model for Tests of Avionics Embedded Systems*. In: Modelsward, Barcelona, Spain, 2013.

[17] D. Gasevic, D. Djuric and V. Devedzic. *Model Driven Architecture and Ontology Development*. Springer-Verlag, Berlin, 2006.

[18] T. Gruber. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Int. Journal of Human-Computer Studies, vol. 43, pp. 907-928, 1995.

[19] S. Weissleder. *Test Models and Coverage Criteria forAutomatic Model-Based Test Generation with UML State Machines*. Humboldt-Universität zu Berlin, Berlin, 2010.

[20] T. Moser, G. Düee and S. Biffl, *Ontology-Based Test Case Generation For Simulating Complex Production Automation Systems*. In: SEKE 2010, San Fransisco Bay, USA, 2010.

[21] C. Nguyen, A. Perini and P. Tonella. *Ontology-based Test Generation for Multiagent Systems.* In: 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Estoril, Portugal, 2008.

[22] B. Zeigler. *Multifacetted Modelling and Discrete Event Simulation.* Academic Press Professional, Inc., 1984.

[23] B. Zeigler, H. Praehofer and T. Kim. *Theory of Modeling and Simulation: Integrating discrete event and continuous complex dynamic systems*. Academic Press, Inc., 2000.

[24] M. Traoré and A. Muzy. *Capturing the dual relationship between simulation models and their context.* Simulation Modelling Practice and Theory, 14, pp. 126-142, 2006.

[25] D. Foures, V. Albert and A. Nketsa. *Simulation Validation Using the Compatibility between Simulation Model and Experimental Frame.* In: 45th Summer Simulation Multi-conference, Toronto, Canada, 2013.

[26] K. Holger, M. Horridge, M. Musen, A. Rector, R. Stevans, N. Drummond, P. Lord, N. Noy, J. Seidenberg and H. Wangl. *The Protege OWL Experience.* In: OWLED, Galway, Ireland, 2005.

[27] P. Perfect, E. Timson, M. White, R. Erdos, A. Gubbels and A. Berryman. *A Rating Scale for Subjective Assesment of Simulator Fidelity*. In: 37th European Rotorcraft Forum, Gallarate, Italy, 2011.

[28] ICAO, *Manual Criteria for the Qualification of Flight Training Devices*. ICAO, Quebec, Canada, 2009.

[29] RAeS, *Aeroplane Flight Simulation Training Device Evaluation Handbook Vol.1 Objective Testing*. RAeS, London, 2009.

[30] B. Nader and J. B. Filippi, *An Experimental Frame for the Simulation of Forest Fire Spread*. In: Proceeding of the 2011 Winter Simulation Conference, Phoenix, Arizona, USA, 2011.

[31] A. Zengin and M. Ozturk, *Formal verification and validation with DEVS-Suite: OSPF Case study.* Simulation Modelling Practice, vol. 29, pp. 193-206, 2012.

[32] O. Corcho and A. Perez, *Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages*. In: ECAI'00 Workshop on Applications of Ontologies and Problem Solving Methods, Berlin, Germany, 2000.

[33] W. Borst, J. Akkermans, A. Pos and J. Top. *The PhysSys Ontology for Physical System*. In: QR'95 Ninth International Workshop on Qualitative Reasoning, Amsterdam, Netherlands, 1995.

[34] W. Borst and J. Akkermans. *Engineering Ontologies.* International Journal of Human-Computer Studies, vol. 46 (2/3), pp. 365-406, 1997.

[35] P. Fishwick and J. Miller. *Ontologies for Modeling and Simulation: Issues and Approaches*. In: Winter Simulation Conference, Washington, DC, 2004.

[36] U. Durak, H. Oguztuzun and K. Ider. *Ontology Based Trajectory Simulation Framework*. Journal of Computing and Information Science in Engineering, vol. 8(1), March 2008.

[37] U. Durak, H. Oguztuzun and K. Ider. *Ontology Based Domain Engineering for Trajectory Simulation Reuse*. Journal of Software Engineering and Knowledge Engineering, vol. 19(8), December 2009.

[38] B. Zeigler. *Modeling & Simulation-Based Data Engineering: Introducing pragmatics into ontologies for net-centric information exchange*. Academic Press, Inc., 2007.

[39] B. Zeigler and H. Sarjoughian. *Guide to Modeling and Simulation of Systems of Systems*. Springer, 2013.

[40] I. Harrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C, Canada, 2004.

Development of a Container Terminal Simulation Ontology

Ann-Katrin Lange[1*], Giovanni Pirovano[2], Rosella Pozzi[3], Tommaso Rossi[3]

[1] Institute of Maritime Logistics of Hamburg University of Technology, Schwarzenbergstraße 95 D, 21073 Hamburg, Germany; * *ann-kathrin.lange@tuhh.de*

[2] Dept. of Management, Economics and Industrial Engineering of Politecnico di Milano University, Via Lambruschini 4/b, Milan, Italy

[3] Industrial Engineering School of LIUC – Cattaneo University, C.so Matteotti, 22 - 21053 Castellanza, Italy

Simulation Notes Europe SNE 24(2), 2014, 79 - 86
DOI: 10.11128/sne.24.tn.10243
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.);
Revised: Oct. 20, 2014; Accepted: October 30, 2014;

Abstract. This article introduces a simulation ontology to support terminal planers, operator and managers in the design and management of seaport container terminals. Due to the increasing requirements of shipping companies regarding efficiency, quality and price for the handling processes at container terminals, the use of integrated approaches for improving the performance has grown significantly. Simulation, which has proven highly beneficial in production and logistics, represents an adequate tool to deal with complex systems like container terminals. However, building simulation models requires much time and simulation software know-how. To counteract this effect, this article presents a simulation ontology of seaport container terminals, which supports the user in building specific simulation models.

Since the simulation model is automatically created through the ontology framework, neither the personnel skills nor the time available to build the simulation model represent significant hurdles. Furthermore, the proposed ontology can dramatically reduce the time required to test a specific configuration of a container terminal and/or a particular management policy. The ontology framework consists of a user interface with database, where the user can specify elements and their parameters, an atom library representing all elements of the system and software application, which is used to automatically build the simulation model.

Introduction

Around 80 per cent of global merchandise trade by volume were transported by water and therefore handled in ports in 2012 [1]. This fact highlights the further growing strategic economic importance of seaborne transport. Combined with low and at the same time volatile freight rates for maritime transport, still remaining after the economic crisis in 2008, and rising bunker prices, this development stimulates the ongoing increase of ship dimensions. This derives a special significance for container vessels, which transport around 52 per cent of global seaborne trade in terms of value [2].

The size of the largest container vessels has more than doubled from around 8,000 twenty-foot equivalent units (TEU) in 2004 to over 18,000 TEU in 2013. Without optimization of the container handling, this would lead to much longer berthing times for unloading and loading the vessels and, as a consequence, to sharp rising port charges. But due to the weak bargaining position of container terminals in relation to shipping companies [3], the terminal operators are pressed to optimize their efficiency and productivity while keeping up low prices. Therefore, all operations and services at the container terminal have to be evaluated and, if necessary, redesigned and adjusted with high investments to meet the stringent demands of a higher turnover in short time windows and higher quality [4].

Seaport container terminals can be considered, in term of material flow, as open systems with two interfaces to other linked systems [5].

One of the interfaces is the quayside, where container ships are assigned to a specific berth and discharged and charged by a set number of ship-to-shore cranes. The other interface is the landside with the unloading and loading of trucks and trains, which can be carried out be different kinds of internal equipment, e.g. rubber-tired-gantry cranes or straddle carriers. The transport of containers from the interfaces to the stocking yard or vice versa is carried out by horizontal transport means, which may differ depending on the required task.

As a result, there is a big variance in the used internal equipment in all areas of the terminal, depending on many different factors, e.g. the stocking systems in the container yard, the average and maximum size of the landing container vessel, the labour costs in the area, the available space in the port, security requirements and the demanded productivity [4;5;6].

Recent developments are e.g. the increasing automation of handling processes to reduce labour costs and to optimise quality and using advanced spreaders to lift multiple containers for enhancing the productivity as well as lowering costs.

To plan, analyse, manage and optimise the complex system of a container terminal, it is no longer sufficient to rely on the knowledge of singular experts or on the problem solving competence of departments in specific, isolated areas. Therefore, integrated approaches for improving the performance of container terminals have been developed. Apart from analytical approaches, there is a focus on simulation based approaches [4], which have proven highly beneficial as decision support systems in production and logistics in general [7] and for container terminals in particular [8;9;10].

For newly planned container terminals, simulation models can provide a preview of the expected overall performance and support the identification of problems before implementing the system. For already existing container terminals, simulation can help to identify bottle necks and optimisation potentials in the current situation and compare them to alternative operation approaches, which can be tested easily and without risks.

This article introduces a methodology to support decisions of seaport container terminal planners, operators and managers concerning the terminal layout, equipment and operations by developing a simulation ontology.

This simulation ontology overcomes the limitations of usual simulation models whose building requires much time and simulation software know-how of the user. Instead it enables the user to quickly build specific simulation models by simply entering all relevant characteristics of the seaport container terminal and its equipment in a user friendly interface.

The article is structured as follows. Section 1 provides information on the state of research about simulation models of container terminals on the one hand and about simulation ontologies on the other hand. Section 2 presents the proposed methodology for the simulation ontology framework for container terminals by explaining the architecture, the atoms library and the software application. Finally, some concluding remarks and suggestions for future research are described in Section 3.

1 Background

Seaport container terminals are complex systems because of several reasons. First of all, there exists a wide variety of organisational forms in regard to terminal operations and used equipment. Many factors have to be considered: How is the layout of the terminal? Which modes of transportation are connected to the terminal? What kind of vessels can be discharged and charged (e.g. maximum size)? Is there a freight station? What kind of equipment is used? Are some processes automated? Are there special areas for storing empty containers, dangerous goods or reefer containers?

Second, after evaluating the organisational form of the terminal, many decisional variables have to be considered, e.g. the number of every kind of equipment and its capacity and the speed of horizontal and vertical transport. In addition, many of these variables may be linked and, as a consequence, influence each other.

Third, static constraints have to be considered as well as dynamic ones. Static constraints are e.g. the number of bays available for the landing of container vessels or the direction of roads on the terminal. Dynamic constraints are for example the work schedules of the staff, the repair schedules of the equipment and the arrival times of ships, trains and trucks at the container terminal.

Fourth, extreme weather conditions and failures of the equipment represent sources of uncertainty, which have to be taken into account.

Because of the complexity, modelling a whole container terminal analytically has proven a challenge [10;11] and the popularity of simulation models for this task has grown significantly [e.g. 6;12;13]. Many of the simulation models focus on one specific area of operation of a container terminal e.g. automated container yard blocks or automated container terminals [14;15], management of berth crane operations [16;17] or the analysis of horizontal transport means [18]. Furthermore, there are some simulation models representing one whole, specific container terminal [11;19].

Although simulation models are important parts of industrial system analysis and control system design, the simulation design issue, the fact that manual simulation design is known as time consuming and error-prone work, has not yet been solved [20].

As a way to overcome such an issue, literature has turned to ontologies, which are a way of formalizing knowledge in a machine-understandable form. In detail, ontologies can be defined as a collection of the kinds of entities that exist in a domain (an identified system), their properties, and the relationships that can hold between them [21]. Ontologies also deal with concepts as ontology classes and individuals, i.e. instances of ontology classes.

Novák and Šindelář formalize knowledge on large-scale industrial systems and use a semi-automated semantic engine that assembles the simulation model, introducing a significant impact of using ontology-based methods on simulation model design phase [20].

Miller et al. develop an example of an ontology for discrete-event modelling, which is a very general domain, identifying the concepts that are most relevant for the discrete-event modelling domain, the relationships between them, the overall architecture, and some of the technical steps involved in creating, deploying and using such an ontology [22].

The present project aims at developing the ontology framework of container terminal simulation models. The individual simulation model of a system associated to the container terminal, which the ontology refers to, is given by a particular instance of the ontology class [20;21;22], similarly to the individual simulation model which is a particular instance of the ontology it refers to [20;23;24]. This instance is automatically obtained based on user-specified input data.

2 The Proposed Methodology

The container terminal simulation ontology framework presented in this paper is based on the simulation software package 'Enterprise Dynamics' (ED) of Incontrol Simulation Solutions.

In the following sections, the overall architecture of the simulation ontology (Section 2.1), the relevant objects (2.2) and the software application (Section 2.3) are presented.

2.1 Ontology architecture

The ontology allows the container terminal manager both to define by means of classes the topology, the resources and the characteristics of the real container terminal under study and to automatically build the corresponding simulation model. By experimenting on such a model, the container terminal manager can verify in advance the performance of the terminal and makes decisions to improve such performance.

The ontology framework involves (see Figure 1): (1) a user interface with database; (2) an ad hoc objects library; (3) a software application. Through the user interface, the container terminal manager specifies both elements and elements' characteristics (the values of the parameters the elements are characterised by) of the container terminal: the values manually entered by the terminal manager and the ones calculated by the same interface are recorded into the database.

The ad hoc library contains both ED atoms and specifically conceived atoms that represent the building blocks of a container terminal: each atom is described by data, represented by the atom attributes, and behaviour, modelled by 4Dscript code (4DScript is the programming language ED is based on). The atom attributes are the parameters that characterise the corresponding container terminal element, while the behaviour is the simulation sub-model, which represents how the corresponding element behaves.

Finally, the software application allows the simulation model to be automatically built. From the database, the application reads the elements of the container terminal and, for each of them, it:

- (1) selects the corresponding atom from the ad hoc library;

- (2) selects from the database the values to be assigned to each atom attribute;
- (3) makes the assignments;
- (4) inserts into the simulation environment the (parameterised) atom.

Once the software application has completed the above-mentioned steps for all the elements of the container terminal, the simulation model is built and the terminal manager can experiment on it.

The atoms library and software application are presented in sections 2.2 and 2.3, respectively. There is no section devoted to either the database or the user interface: actually the database consists of a standard ED table atom, while the interface is given by mere data-entry masks, which allow to specify the values of the container terminal elements parameters, which are the mirror image of the objects data.

2.2 Atoms library

Two different classes of objects belong to the ad hoc library: (1) the roads, which allow to create the network where the transport means characterising the container terminal move; (2) the resources, which perform the activities within the container terminal: gate, service line, rail crane, active transportation equipment (i.e. straddle carrier, reach-stacker, container lift truck), passive transportation equipment (automated guided vehicles, tractor and trailer, multi-trailer), stacking yard and portal crane.

Trucks, trains, ships and containers are not considered as objects since they are the entities which flow along the simulation models which in turn can be automatically built by the simulation ontology framework.

The objects are described by data and behaviour (as pointed out in Section 2.1). As for the data, the attributes related to the atoms of the roads class are: (1) the couple of nodes linked by the road; (2) the parameters required to set possible constraints (direction, number of driving lines, traffic rule and speed factor).

The attributes related to the atoms of the resources class basically quantify the terminal resources occupancy (storage capacity, cranes rate, inter-arrival traffic time at piers, etc.).

As for the behaviours, they are the simulation sub-models which represent how the container terminal elements interact with each other and with the entities represented by trucks, trains, ships and containers. To provide an example, Table 1 presents the data and behaviour (i.e. the attributes and the simulation sub-model) of the object (i.e. atom) 'active transportation equipment' (in particular, due to the functioning of ED, the sub-model is represented by means of state machine diagrams and attributed Petri nets).

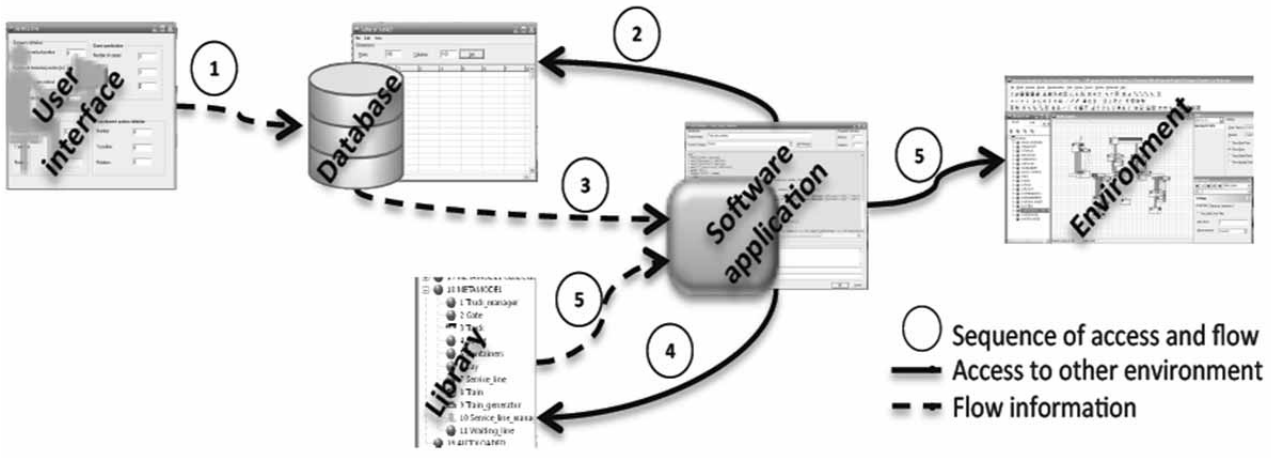


Figure 1. Ontology architecture.

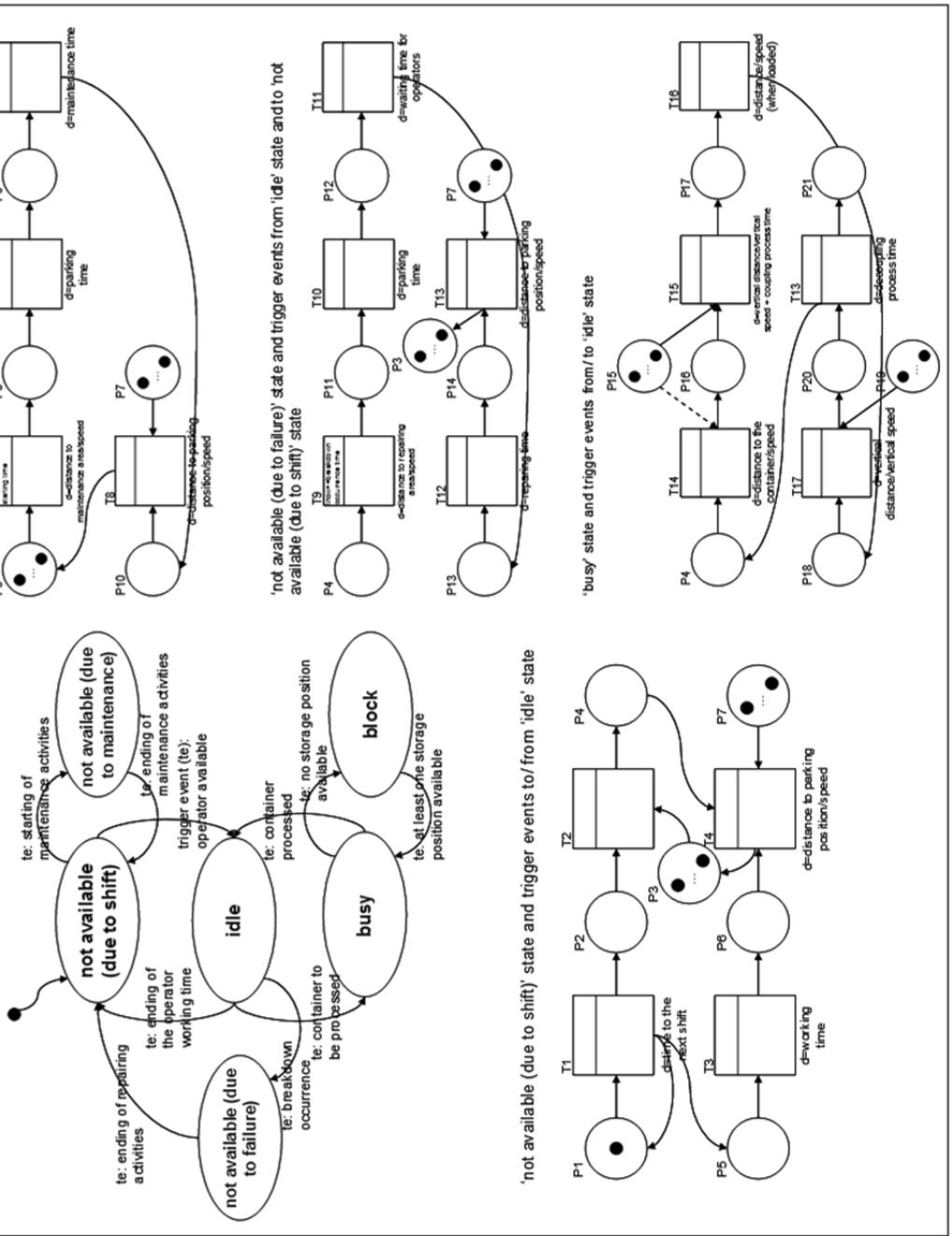
Object	Data	Behaviour
Active transportation equipm	*Capacity* [# of container that can be handled together]	
	Type of container the equipment can handle	
	Cycle times (coupling process, decoupling process, etc.) [s]	
	Speeds (straight, corner, block, loaded, etc.) [m/s]	
	Stacking high [m]	
	Dispatching rules	
	MTBF	
	MTTR	
	Shift and break schedule	
	Network (where the transportation equipment moves)	

Table 1: Data and behaviour of the object 'active transportation equipment'.

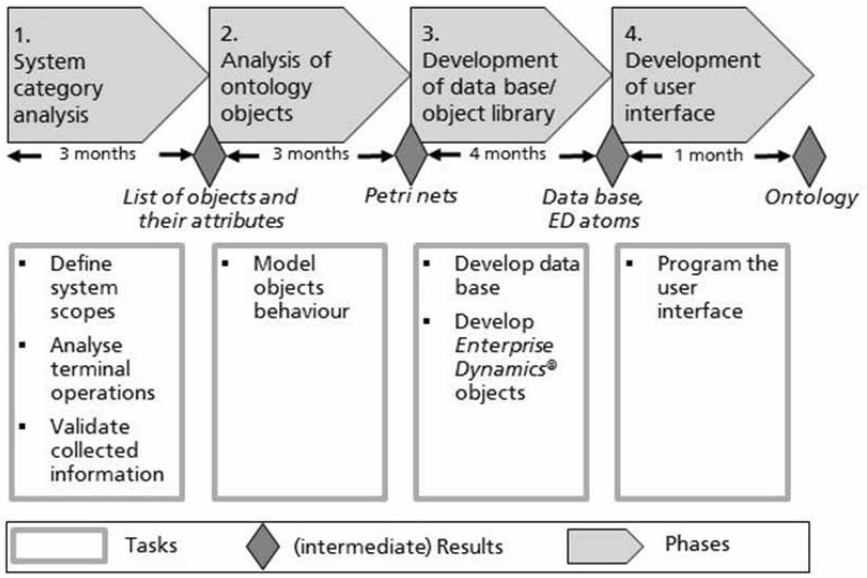


Figure 2: Project phases for container terminal ontology framework development

2.3 Software application

The software application is divided in two sub-procedures: the first one inserts into the simulation environment the atoms representing the objects belonging to the resources class; the second creates the network by inserting a road atom between all the couples of resource atoms, which represent elements that are included into the real container terminal network.

The first sub-procedure starts by accessing the database, i.e. the table atom, and by reading all the parameters related to the first object belonging to the resources class, which must be created. Then the software application accesses the ad hoc atom library and inserts into the simulation environment the corresponding atom. After that, the application assigns the previously red values to the specific atom.

This sub-procedure is repeated for all the resources objects defined by the user. When all the resources objects have been created, the first sub-procedure ends and the second one starts, in order to create the network. In particular the sub-procedure reads again from the table atom the parameters (allowed speed, number of driving lines, etc.) that characterize the link between the first couple of resources objects.

Then the software application chooses the road atom from the ad hoc library, inserts it into the simulation environment and assigns to the attributes of the atom a alue according to the previously red parameters.

At that time the second sub-procedure stops, since the network is completely modelled into the simulation environment, the simulation model of the container terminal under study is ready to be used.

3 Conclusion and Outlook

This article is focused on the development of an ontology to support the design and the management policies definition of a seaport container terminal.

The reason to address this problem is given by the fact that, despite simulation is considered one of the most promising tools to support the design and to manage container terminals; its use in real-life contexts is limited by the high requirements for development time and simulation know-how.

To address the problem a joint project is carried on by the Institute of Maritime Logistics of Hamburg University of Technology – MLS (Hamburg, Germany) and Cattaneo University – LIUC and Politecnico di Milano University. The project is structured in four phases (see Figure 2). Currently, phase 3 is in progress.

The first phase has dealt with the deep analysis of the category of systems which the simulation ontology refers to, i.e. container terminals. A set of questions have been asked to practitioners to obtain necessary information. In addition, inspections to exemplar maritime container terminals in Hamburg and Bremerhaven have been performed. Literature has been used to gather further input from other container terminals and to consider all well-known established and planned types of terminal operation systems.

The output of the first phase has been identified by a list of all the objects the simulation ontology must include and the attributes characterizing each object (see paragraph 3.1. For an example of attributes, see column 'Data' of Table 1).

In the second phase, the logical model representing the behaviour of each listed object has been developed. In order to facilitate the development process and to ensure the completeness of the logical model a state machine diagram has been constructed for every object. This diagram displays all the states that the object can visit, the trigger events for changing into another state and the action performed before and after the state change. Based on these state machine diagrams the Petri nets formalism has been used to display the internal behaviour during the different states and the behaviour during a state change (for an example, see table 1). The developed state machine diagrams and Petri nets identify the output of the phase.

The third phase of the project, which is in progress, consists of the development of the database and of the ad hoc object library by means of ED. So far, the database has been developed by means of the ED atom 'Table'; the ad hoc atoms representing the objects 'road', 'service line', 'rail crane' 'stacking yard', 'tractor and trailer' have been developed in 4DScript; a beta version of the software application to automatically build the simulation model has been coded in 4DScript.

In the last phase of the project a user interface will be developed by means of the ED application 'GUI builder'. This interface will allow the user to define the elements composing the container terminal which shall be simulated as well the container terminal topology. Furthermore, it will allow the user to easily enter the necessary values into the database to define the attributes of each system element.

The user interface will be built so as that, on the one hand, the user can decide for many of the attributes if he/she wants to enter values or use the deposited ones and, on the other hand, the input of data can as well be done by loading complex lists, e.g. ship arrival times, as by filling out dialog windows. The completed container terminal ontology framework will identify the output of the last phase.

The main strengths of the proposed simulation ontology can be summarised as follows: first, it represents a specifically conceived decision support tool for solving an optimisation problem under several constraints, uncertainty and many interdependent variables as the design or the definition of management policies of a container terminal is. Second, the proposed ontology allows to take into account some dynamic features of the terminal system unmanageable by manual calculations (e.g. the arrival times of ships, trains and trucks at the terminal).

Finally, the time required to test a specific configuration of the system is dramatically reduced: in a few minutes the user can specify the elements the system is composed of as well as the elements' characteristics through the user interface; the software application builds the corresponding simulation model, which can be run, depending on the simulation length and on the hardware, in a few seconds or in a few minutes, on behalf of the decision-making process speed.

The future of the research line outlined here is oriented towards proving the effectiveness of the proposed simulation ontology framework by using it to estimate the performance of an existing container terminal system: for this reason, a study of the La Spezia container terminal has been already planned.

References

[1] UNCTAD Secretariat. *Review of Maritime Transport.* United Nations Publication, Switzerland, 2013.

[2] World Shipping Council. *Value of world seaborne trade.* See http://www.worldshipping.org/ (accessed 10 July 2014), 2014.

[3] T.-F. Wang and K. Cullinane. *The efficiency of european container terminals and implications for supply chain management.* Maritime Econ Logistics 8, p. 82–99, 2006.

[4] R. Stahlbock and S. Voß. *Operations research at container terminals: a literature update.* OR Spectrum 30, p.1–52, 2008.

[5] D. Steenken, S. Voß and R. Stahlbock. *Container terminal operations and operations research—a classification and literature review.* OR Spectrum 26, p.3–49, 2004.

[6] P. Ghanbari. *Containerterminal-Logistik.* VDM Verlag Dr. Müller, Germany, 2007.

[7] L. März and G. Weigert. *Simulationsgestützte Optimierung.* In: Simulation und Optimierung in Produktion und Logistik, Editor: L. März, W. Krug, O. Rose und G. Weigert, Springer-Verlag Berlin Heidelberg, Germany, p. 3-12, 2011.

[8] R. Gibson, B. Carpenter and S. Seeburger. *A flexible port traffic planning model*. Proceedings of the 1992 Winter Simulation Conference, in USA, p. 1296–1306, 1992.

[9] P.-H. Koh, J. L. K. Goh, H.-S. Ng and H.-C. Ng. *Using simulation to preview plans of a container port operations*. Proceedings of the 1994 Winter Simulation Conference, in USA, p. 1109–1115, 1994.

[10] Y. Merkuryev, J. Tolujew, E. Blümel, L. Novitsky, E. Ginters, E. Viktorova, G. Merkuryeva, J. Pronins. *A modelling and simulation methodology for managing the Riga harbour container terminal*. Simulation 71, p. 84–95, 1998.

[11] W. Y. Yun and Y. S. Choi. *A simulation model for container-terminal operation analysis using an object-oriented approach.* International Journal of Production Economics 59, p. 221-230, 1999.

[12] L. M. Gambardella, A. E. Rizzoli and M. Zaffalon. *Simulation and planning of an intermodal container terminal.* Simulation 71(2): p. 107–116, 1998.

[13] T. Franz. *Marktbasiertes Containerterminal-Management - Grundlagen, Mechanismen und verteilte Simulation.* VDM Verlag Dr. Müller, Germany, 2008.

[14] N. Kemme. *Effects of storage block layout and automated yard crane systems on the performance of seaport container terminals*. OR Spectrum 3, p. 563-591, 2012.

[15] P. Canonaco, P. Legato, R. Mazza and R. Musmanno. *A queuing network model for the management of berth crane operations*. Computers & Operations Research 8, p. 2432-2446, 2008.

[16] C.-I. Liu, H. Jula and P. A. Ioannou. *Design, Simulation, and Evaluation of Automated Container Terminals*. IEEE Transactions on intelliegent transportation systems 3 no. 1, p. 12-26, 2002.

[17] J. Dai, W. Lin, R. Moorthy and C.-P. Teo. *Berth allocation planning optimization in container terminal*. Working paper, Georgia Institute of Technology, Atlanta; National University of Singapore, 2004.

[18] M.B. Duinkerken, R. Dekker, S.T. Kurstjens, J.A. Ottjes and N.P. Dellaert. *Comparing transportation systems for inter-terminal transport at the Maasvlakte container terminals.* OR Spectrum 28, p. 469–493, 2006.

[19] A.A. Shabayek and W.W. Yeung. *A simulation for the Kwai Chung container terminal in Hong Kong*. European Journal of Operational Research 140, p. 1–11, 2002.

[20] P. Novák and R. Šindelář. *Applications of ontologies for assembling simulation models of industrial systems.* On the Move to Meaningful Internet Systems: OTM 2011 Workshops. Springer Berlin Heidelberg. p. 148-157, 2011.

[21] P. Benjamin, M. Patki and R. Mayer, *Using ontologies for simulation modeling.* In: Proceedings of the 38th conference on Winter simulation. Winter Simulation Conference, p. 1151-1159, 2006.

[22] J. A. Miller, G. T. Baramidze, A. P. Sheth, G. A. Silver and P. A. Fishwick, *Ontologies for modeling and simulation: An initial framework.* Computer Science Department, University of Georgia, Athens, GA, 30602, 2004.

[23] R. Cigolini and T. Rossi. *Sizing off-shore transshipment systems in dry-bulk transportation.* Production Planning & Control 21, p. 508-522, 2010.

[24] R. Cigolini, M. Pero and T. Rossi. *An object-oriented simulation meta-model to analyse supply chain performance*. International Journal of Production Research 49, p. 5917-5941, 2011.

[25] V. Boschian, M. Dotoli, M.P. Fanti, G. Iacobellis and W. Ukovich. *A Metamodeling Approach to the Management of Intermodel Transportation Networks.* IEEE Transactions on Automation Science and Engineering 8 no. 3, p. 457-469, 2011.

OverNight Testing – The Fully Automated Simulation Environment for Evaluation of Car Concepts ONT

Mark Krausz[1*], Matthias Zimmer[2], Hans Christian Reuss[1]

[1] Research Institute of Automotive Engineering and Vehicle Engines Stuttgart (FKFS), Pfaffenwaldring 12, 70659 Stuttgart, Germany, *mark.krausz@fkfs.de*

[2] Dr. Ing. h.c. F. Porsche AG, Porschestr 911, 71287 Weissach, Germany

Simulation Notes Europe SNE 24(2), 2014, 87 - 94
DOI: 10.11128/sne.24.tn.10245
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.);
Revised: Oct. 20, 2014; Accepted: October 30, 2014;

Abstract. In this paper, a short introduction of the full vehicle simulation environment OverNight Testing (ONT) is given. This environment was developed in cooperation of Dr. Ing. h.c. F. Porsche AG with the Research Institute of Automotive Engineering and Vehicle Engines Stuttgart (FKFS) [1]. The main application area is in the concept development for the assessment of overall vehicle concepts. The distinctive features of the simulation environment are a strong configurability and the high degree of automation and modularization.

A process has been defined for the evaluation of a car concept, starting with the integration of new model components continuing with the creation of a new vehicle configuration, over the simulation using various manoeuvres up to the presentation of results. The advantage of the presented simulation tool is the ability to handle a huge number of variaties of a vehicle concept automatically and create evaluation results quickly by using the toolkit principle.

Introduction

The number of vehicle derivatives is increasing in the automotive industry for years. With the introduction of platforms and the toolkit principle these variants must stay manageable [2]. A large number of variants for each car concept need to be evaluated within a short time.

Hence it is the goal to use full vehicle simulation models with the same modular structure as the toolkit principles to make the diversity and complexity controllable [3]. The vehicle model should be divided into modules that can be replaced and are reusable. A very important additional constraint is, that the amount and detail of information in the early phase of the vehicle development process are low [4].

The models of the vehicle components with different levels of detail are combined to a complete full vehicle model. In the course of the project the simple models can be easily replaced by more complex and larger models. A clearly defined and structured evaluation process is essential to make the high number of evaluations manageable [5]. With the help of a developed domain ontology [6] the automated linking of the various forms of information, objects and properties is made feasible.

Furthermore, it must be possible to evaluate the vehicle models with a variety of technologies as well was parameter sets and comparing the results. Easy and rapid variations of the components and the parameters of the model are crucial in the concept evaluation. It is of high importance to make the expertise of other disciplines available by using and exchanging models over simulation platform boundaries [7].

Therfore the model boundaries and interfaces need to be cleary defined. A user-friendly interface will be provided to review the results easily and quickly. All data used in the evaluation and the results should be stored to be able to repeat the review and understand the results anytime later.

1 The Simulation Environment OverNight Testing (ONT)

The prototype of the simulation environment ONT implements the above mentioned requirements. The Starting window of ONT is depicted in Figure 1. The tool was created in MATLAB and Simulink and enables a simple and user-friendly simulative evaluation of vehicle concepts.

MATLAB and Simulink was chosen on the one hand due to the possibilities for creating graphical user interfaces, easy implementation and integration of functions for processing data, which are necessary for the pre- and postprocessing of simulations and lastly as the program is widely used for the simulation of vehicle behavior within the company. The user does not need prior experience with MATLAB or Simulink since all functions can be operated via user interfaces.

Figure 1. ONT starting window.

The following is an example for a concept evaluation which will be explained gradually.

1.1 Example for a concept evaluation

In this example, the fictional concept of an electric vehicle with the name *Concept_1* is evaluated. The standard procedure for the review of a vehicle in ONT is shown in Figure 2.

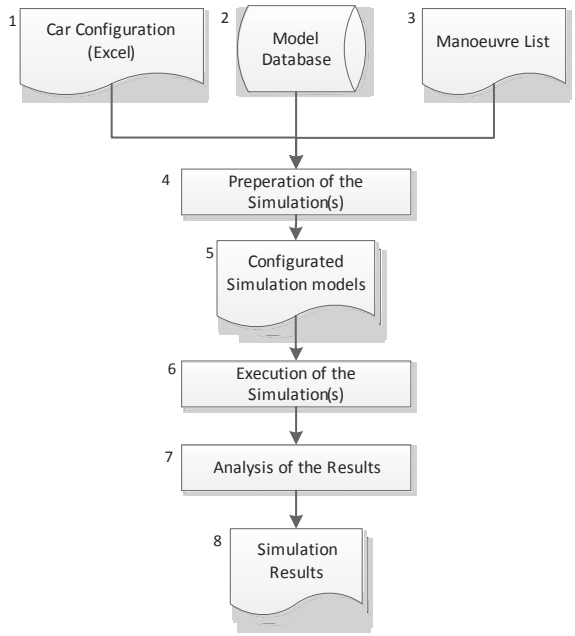


Figure 2. Standard evaluation procedure in ONT.

In order to evaluate a vehicle concept with the simulation tool ONT the name of the components must be stored in an Excel file (1) in a predefined hierarchical structure. The information in the Excel file is imported and can be seen in extracts in Figure 3.

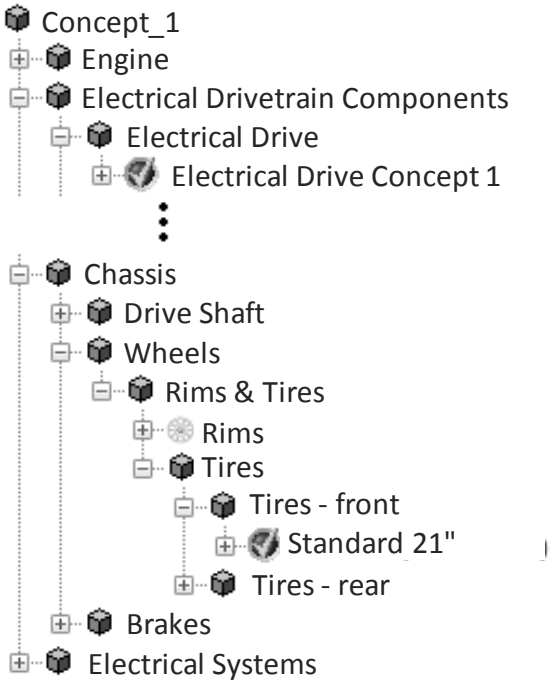


Figure 3. Extract from the tree view of the vehicle configuration in ONT.

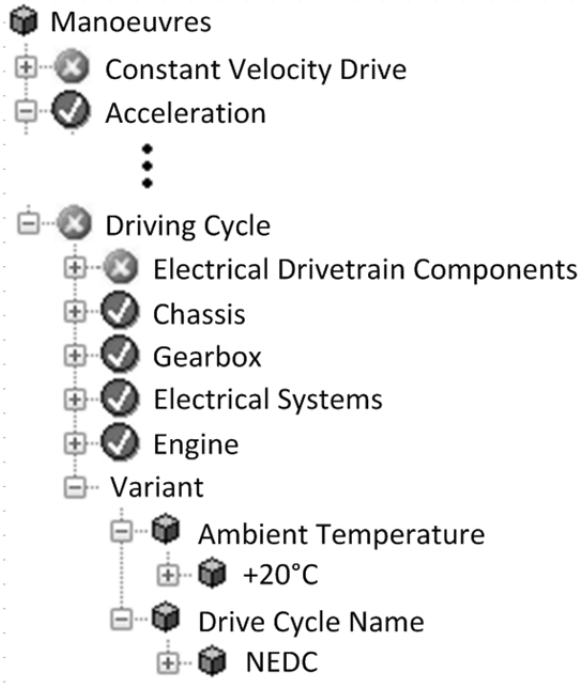


Figure 4. Example of a manveuer tree view with its requirements and variants

Not only the information about the components of the vehicle has to be defined, also the simulation-scenarios (in this paper called manoeuvres) have to be considered. Therefor a list (3) can be prepared consisting of pre-defined manoeuvres. It is possible to simulate e.g. different types of acceleration, drives with constant velocity, elasticity, driving cycles and circuits. Boundary conditions and requirements, like ambient temperature, SOC values and speed limits are given by the user.

In Figure 4 an extract of the tree view of the manoeuvre is shown. The manoeuvres have requirements on each component of the full vehicle model that are fully automated verified.

These requirements are shown in the same hierarchy as the vehicle structure in a tree view beneath every manoeuvre. If a requirement is met a check mark shows up. If it is not met the components and its higher-level category is flagged with a cross.

Once a component is marked with a cross the manoeuvre cannot be simulated. The requirements on the manoeuvres are described in more detail in section 2.6.

In this example only one manoeuvre can be simulated due to the available vehicle data and manoeuvre request. The two manoeuvres, driving cycle and constant travel, cannot be selected because the requirements to the electric drive components are not met. For example this can be the case if the efficiency of the electrical motor is unknown. An assertion of consumption would be very inaccurate and would not meet the underlying requirements.

From the vehicle definition (2) and the list of manoeuvres (3) fully configured vehicle models are generated automatically with the help of a database in which all the models and parameters are stored (5).

Subsequently, the simulations can be carried out (6). A small status window provides an overview of the current status. The simulation model is hidden from the user in the background.

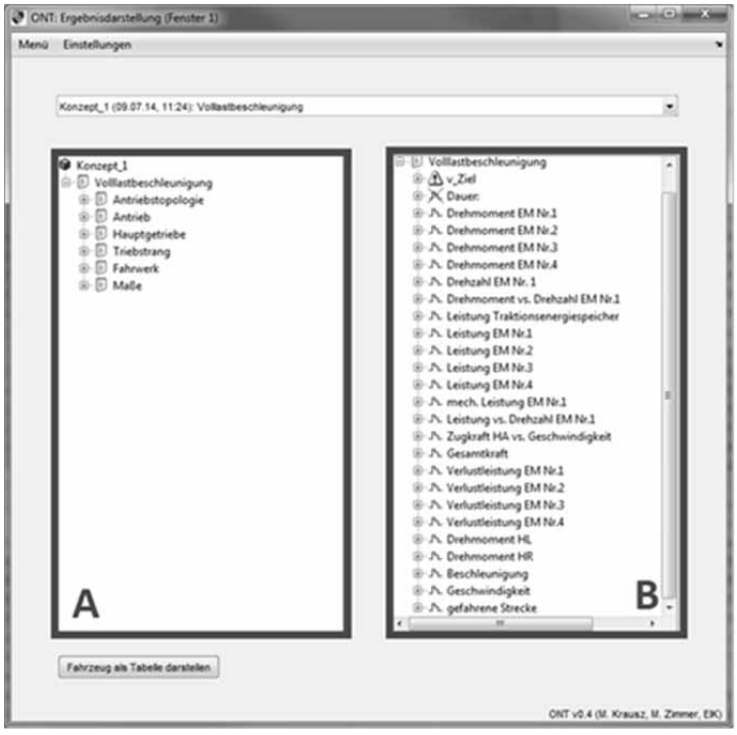


Figure 5. The results window in ONT.

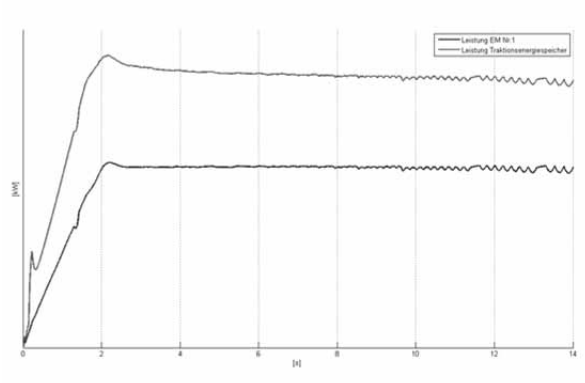


Figure 6. Example for a combined result diagram.

After the simulation, an automated evaluation of the simulation results is optionally performed (7). To view the simulation results, the presentation of results can be opened. In addition to the current former performed simulation results (8) can be viewed anytime. The content of the window, which is shown in Figure 5, is fully configurable.

Besides of the vehicle configuration (A) signals from list (B) can be selected and illustrated in different diagrams. An example of displaying multiple signals in one diagram is shown in Figure 6. The set of shown signals can be configured for each manoeuvre by the user. The signals can be combined in plots and be exported to Excel files for further use.

2 The Vehicle Structure

The mentioned Excel file (see section 2.1) is the underlying hierarchical structure in ONT and will be discussed in more detail in the following. This structure is used throughout the company to list the components integrated in the vehicle. In addition to a folder-based database, in which the models and parameter files are stored, the structure is found again in the signals of the simulation model.

The full vehicle model has a central signal bus which is divided into the same levels. This only serves to structure the variety of signals in a reasonable and continuous way and has no properties of a real communication bus. By reusing this system, models and sets of parameters can be easily found and stored in the file system. Furthermore parts of the full vehicle model and selected individual signals of partial models from the signal bus can be located easily.

Figure 7 is an example to see where the traction energy storage is arranged in the vehicle hierarchy.

2.1 The full vehicle model

The full vehicle model in ONT consists of a variety of components which are represented by different detailed models depending on the manoeuvres to be simulated. Simulations in ONT are always forward simulations. The model at the highest level is shown in Figure 8. All blocks have the full vehicle bus as input. The output of each block is a bus with the aggregated signals of its subcomponents. In section 2.4 the exact buildup of the models is discussed in more detail.

The powertrain of the full vehicle model consists of one or more engines/electric motors, clutch, transmission, power divider, differentials, drive shafts, brakes, wheels and tires. In addition to the physical mapping of components controls and regulations of the systems are also modeled, if necessary. In case of an electrified powertrain the high-voltage aspects are represented by electric drives, power electronics, a traction battery and a charging unit. Through this a very good reproduction of the longitudinal dynamics can be achieved for combustion-engined as well as electrified vehicles.

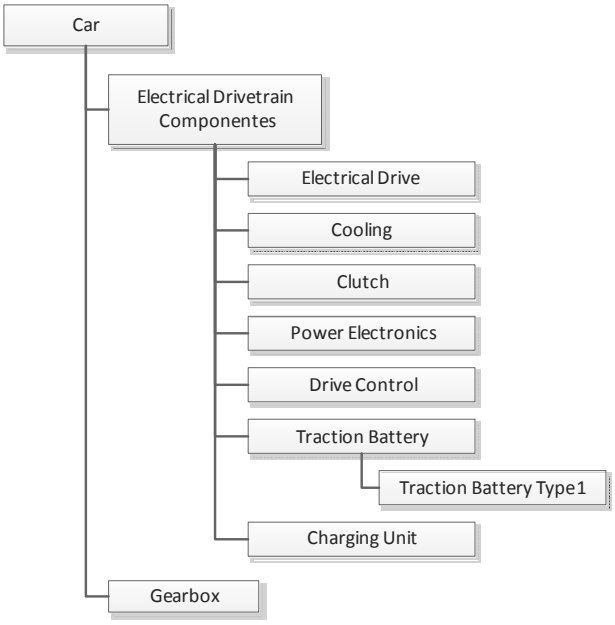


Figure 7. Extract of the tree view of the vehicle hierarchy.

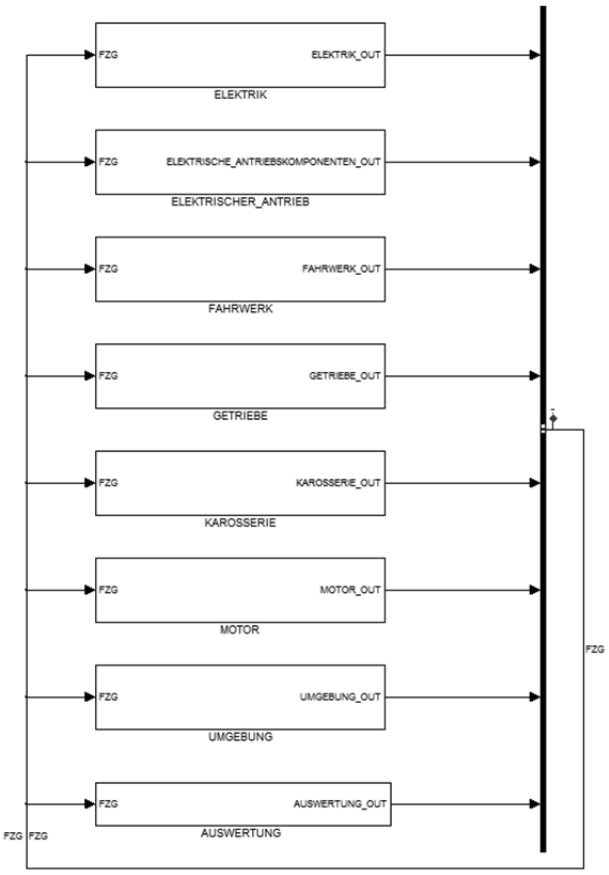


Figure 8. The full vehicle simulation model in Simulink.

2.2 The Models oft the Car Components

Every component used in the full vehicle simulation is filed in the same way. A component can be modelled in different ways and therefore be represented by different models. Simple models are used, if there is only little information about the system available. Over time the amount of information is increasing and therefore more complex models can be used. These more complex models need more data to be parameterized.

The level of detail for every variant of the model needs to be defined in five categories. These categories are: mechanics, electrics, thermodynamics, chemistry, and logic. The range reaches from 0 (not modelled) to 10 (reality). In Table 1 an example is shown. It is the excerpt for the definition of the level of detail for the component traction battery.

LoD	Electrics	Thermodynamics
0	Not modelled	Not modelled
1	Simple Resistore	0-dim. System
2	Simple Energy Storage	0-dim. Cell
3	Temp. Dep. Resistor	1-dim System
…		
10	HiL-Battery	HiL-Battery

Table 1. Example for the definition of the level of detail for the model *traction battery*

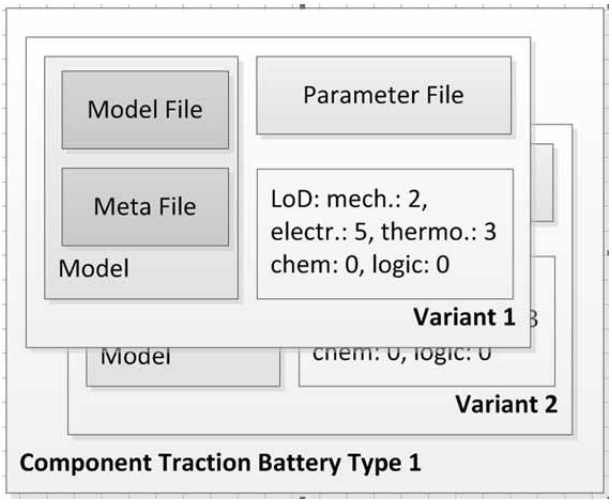


Figure 9. The component *traction battery type 1* with its different parts.

A model variant consists of different parts: The parameter set, the model file itself, and a metafile with additional information about the model. In this metafile in- and output signals, parameters of the model and other information as author of the model, date of creation, restrictions of the model etc. are definied.

In Figure 9 the different parts of the component traction battery type 1 are shown. One extract for the developed domain ontology is shown in Listing 1, which is defined in XML-format.

```
1  <name>…</name> (model name)
2  … (additional informations)
3  <inputs> (list of all input signals)
4        <element>
5        </element>
6  …
7  </inputs>
8  <outputs> (list of all output signals)
9        <element>
10       </element>
11 …
12 </outputs>
13 <parameters> (list of all model parameters)
14 </parameters>
```

Listing 1 Metafile structure for the simulation model of a component

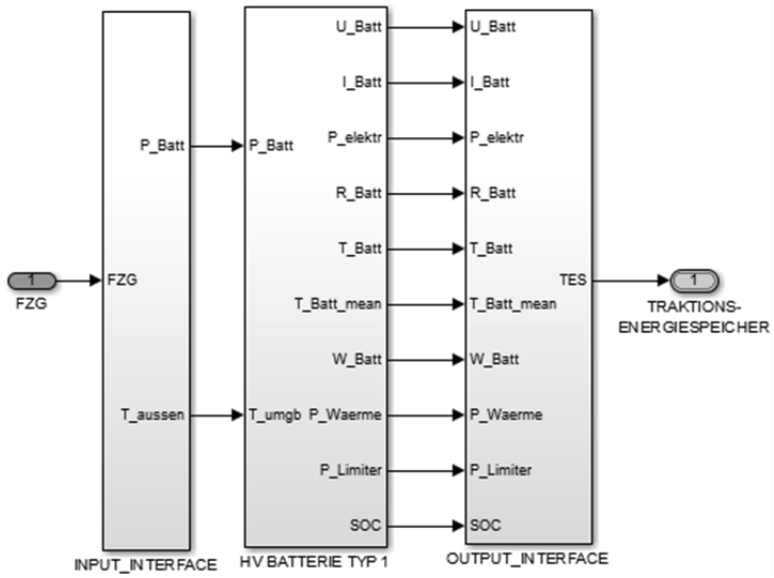


Figure 10. The model *traction battery type 1* with its input and output interfaces.

The simulation model of each component possesses always an input and an output interface. As an example the model of the traction battery type 1 is shown in Figure 10.

The *Input Interface* selects from the full vehicle signal bus all signals, which are definied in the metafile. In this signal bus all signals from every car component model are available. The model output signals are collected in the *Output Interface* block and aggregated in a component-specific signal bus.

With the input and the ouput interfaces models from other departments or suppliers can be integrated easily. To integrate new models only the interfaces need to be adapted and the main model can be remain in its original state.

2.3 The integration of a new component into ONT

A procedure is defined to integrate new compontents or variants of new models into the simultation environment ONT, which is shown in Figure 11.

This process uses different methods of verification and validation [8] that are not focus of this paper and therefore not further explained.

To start the process the new component needs to be defined as shown in section 2.2 with a model file, parameter file, and metafile. In the first step the interfaces of the model are compared with the definitions in the metafile. Beside the correct dimensions and naming the existence of required input signals in the full vehicle signal bus is checked.

Also the parameters of the model are compared to the definitions in the metafile and the additional information of the metafile like solver settings is reviewed. After the formal test procedures the model is installed into a virtual model test bed for the next steps.

This test bed is providing all necessery input signals to run the test and is observing the model behaviour by logging all outputs of the model.

The first step on the test bed is a short test of executability. Afterwards predefined verification tests are run. These verification tests are specifically defined for each component and can consist of simple input-output test and more complex combinded test scenarios. An example for a simple scenario for the traction battery model is to discharge and charge the battery. Another example is to check the model behaviour while the attempt to discharge the battery below 0% SoC (State of Charge).

After all tests are executed the results are saved in a test protocol and added to the component model. The component model is filed to a component library if all tests were passed successfully. Now the component can be used in ONT.

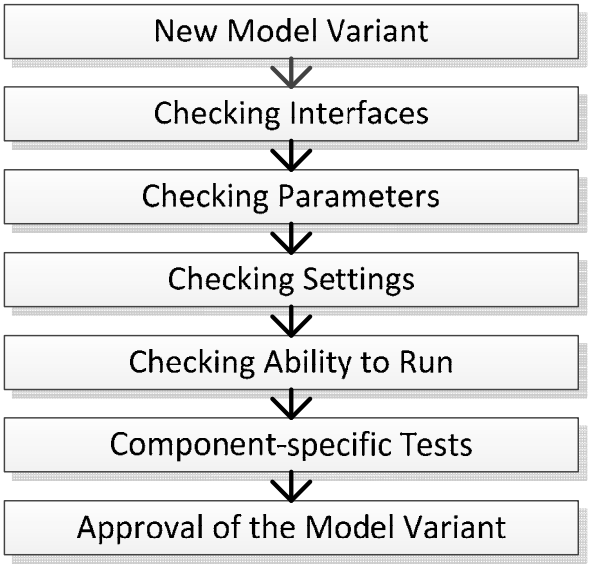


Figure 11. Procedure for the integration of new components in ONT

3 Manoeuvres and their Requirements on the Model

The manoeuvres for the simulation with the full vehicle model are predefined. They can be divided into two main groups:

1. Manoeuvres with time- or distance-based velocity and height profile
2. Event-triggered manoeuvres

The first group of manoeuvres provides a route profile that is traced by a digital driver. As soon as the profile reaches its end the simulation is finished. Examples for this type of manoeuvres are driving cycles, longitudinal driving on a race track, hill climbing, simple accelerations and decelerations.

The second group of manoeuvres depend on events. There is no static profile given, the requested velocity is dynamic.

Examples for this type of manoeuvres are driving until the traction battery is empty, repeated acceleration to a given speed and instant deceleration after reaching it.

The manoeuvres are filed in a database and can be combined with a graphical user interface to a list. Most of the manoeuvres are equipped with some tunable paramters like ambient temperature or starting/finishing speed. Besides the parameters, all manoeuvres have requirements on the level of detail of every component model.

In Section 2.2 the concept of the level of detail for models was introduced. It is used for the preparation of the simulation model. The full vehicle simulation model with all its component models is defined with the information about the manoeuvre and the components of the car concept. To make this possible the concept of manoeuvre requirements are developed. Every manoeuvre defines requirements for each model component of the full vehicle model. These requirements are formulated in level of details in five different categories.

The requirements on the level of detail are mimal requirements that need to be met by each used component model. If none of the available model variants of one single component can fullfill a requirement the manoeuvre cannot be simulated for this vehicle. The idea is to allow simulations only if enough information are available to get trustworthy results.

For each manoeuver a list is definied and saved as a XML-file. This list contains the requirements for every single component specified with the level of detail in five categories. An example for such a manoeuvre requirement list is shown in Figure 12. In this manoeuvre the car drives at constant longitudinal velocity. The aim of the simulation is to calculate the energy consumption. At this manoeuvre the behaviour of the traction battery is important, as this is the energy storage in the car.

On the other side the charging unit of the high voltage system has no influence on the result of the simulation besides its weight and physical size. Therefore the model of the charging unit is not required for this manoeuvre. All required levels of detail are set to zero. With this list of requirements for each model component on the one side and the car configurations on the other side the specific full vehicle model can be defined for each manoeuvre. In this way the most suitable models are used for every simulation.

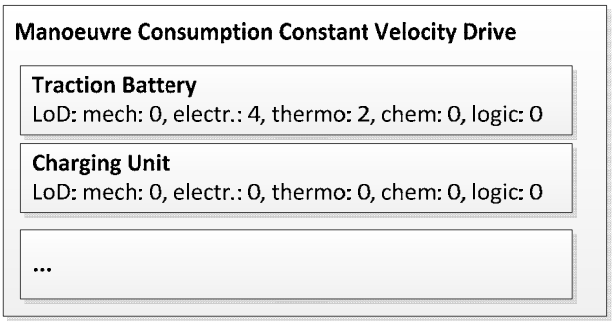


Figure 12. Examplary manuever requirements from the manuever „consumption constant velocity drive"

4 Conclusion and Future Work

The simulation environment ONT is used in the early phase of concept development. A large number of concept variants can be evaluated with it rapidly. This supports the project leaders to make fast and correct decisions. ONT could be used as a simulation platform in all parts of the R&D department, where simulations in many fields e.g. vehicle dynamics are needed.

The structure of ONT follows a standardized and widespread hierarchical system. It is highly configurable and automated. The full vehicle simulation model offers a model architecture, where component models can be included easily. A multidisciplinary project is run in the company to evaluate the benefit of ONT in the R&D department. The results of the evaluation will be shown in future publications.

References

[1] M. Zimmer. *Entwicklung eines durchgängigen Simulationsprozesses zur Effizienzsteigerung und Reifegraderhöhung von Konzeptbewertungen in der Frühen Phase der Produktentstehung*, Dissertation Universität Stuttgart, in publication, 2014

[2] M. Schmieder, S. Thomas. *Plattformstrategien und Modularisierung in der Automobilentwicklung*, Shaker, Aachen, 2005

[3] S. Bewersdorff, J. Pfau. *Beherrschung der Variantenvielfalt durch prozesssichere Simulation*, ATZ Oktober 2011, Volume 113, Issue 10, Springer, Wiesbaden, 774-779

[4] M. Zimmer, M. Krausz, H.-C. Reuss. *OverNightTesting – Ein Simulationsprozess zur Konzeptbewertung in der Frühen Phase*, 7. Grazer Symposium Virtuelles Fahrzeug, 2014

[5] B. Verworn. *Management der frühen Innovationsphasen*, Springer, Heidelberg, 2007

[6] R. Busse, et. al. *Was bedeutet eigentlich Ontologie?*, Informatik Spektrum 37/4 2014, Springer, Heidelberg, 2014

[7] U. Seiffert. *Virtuelle Produktentstehung für Fahrzeug und Antrieb im Kfz*, Vieweg + Teubner, Wiesbaden, 2008

[8] R. G. Sargent. *Verification and Validation of Simulation Models*, Proceedings of the 2003 Winter Simulation Conference, Volume 1, IEEE, Warrendale, 2003

Perdurantist Modeling and Reasoning in Ontology-based Modeling

Mehmat Fatih Hocaoğlu

Istanbul Medeniyet University, Faculty of Engineering & Architecture, Department of Industrial Engineering, Istanbul, Turkey; *mfatih.hocaoglu@medeniyet.edu.tr*

Simulation Notes Europe SNE 24(2), 2014, 95 - 104
DOI: 10.11128/sne.24.tn.10246
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.);
Revised: Oct. 25, 2014; Accepted: October 30, 2014;

Abstract. In this study, the perdurantist modeling approach in which entities have four dimensions (spatial and temporal) and only briefly exist during the different stages of their lifespan is extended by a reasoning mechanism. The extension allows a modeler to manage behaviors depending on reasoning results and also provides well-designed support for time-delayed systems. Language support is provided for this purpose, starting with an ontological commitment and covering design and coding phases, including conceptual model description.
This paper discusses how an agent-driven simulation language supports extending a perdurantist modeling to ontology-based modeling by high-level action descriptions, higher-order world envisionment, dynamic relation management and a knowledge base for reasoning purposed as the necessities of ontology based modeling. One of the study's main aims is to match the predicate logic ontological commitment with the ontological commitment presented here and bring them into a comment framework to handle behavioral management in simulation.

Introduction

Agent-driven Simulation Framework (AdSiF) [1], [2] is a declarative simulation language and a development environment for simulation and agent programming. It basically provides a state-oriented interpreter and a simulation layer to manage simulation execution algorithms for both discrete-event and continuous-event systems.

Compared to current agent programming systems (such as Jason [3]), AdSiF gives a different perspective by fusing the agent-based programming paradigm, object-oriented paradigm, aspect-oriented paradigm [4] and logic programming paradigm [5] into a single paradigm, referred to here as a state-oriented paradigm [1], [6]. State-oriented programming (SOP) was originally introduced in 1987 by D. Harel [7] as a visual formalism to model complex reactive systems. SOP was adopted by OMG in 1997 as a part of UML 2.0 specification and is based on state charts [8]. AdSiF enhances state-oriented programming using multi-programming paradigms and defines it as a programming language.

As a language, AdSiF provides programming by states instead of the programming states as performed in the state charts. It interprets the extended state charts and does not require coding the chart itself. State-oriented programming handles the state transition process, which is declared in the form of the State Charts of AdSiF. In each state, the simulation model spends a certain amount of time to pass through the entire state (or, at least, the model attempts to pass through the whole state) in an orderly fashion, and the simulation models are capable of executing many behaviors in parallel, at any time. The execution of a state-oriented program has a timeline due to the duration that the simulation model spends in each state.

The power of this paradigm stems from its ontological commitment, which extends from describing what exists to include the modeling of mental abilities through the use of a reasoning mechanism, thereby driving behaviors.

From the ontology-based modeling point of view, AdSiF provides a strong programming background to satisfy the fundamental ontological notions, which are identity, unity, rigidity and dependency. Ontologically, AdSiF constructs a world composed of entities capable of managing their behaviors reactively and proactively.

The behavioral aspect of an entity consists of a set of behaviors, which is defined as a sequence of states, a set of events triggering behaviors and behavioral declaration of the relations it has.

A behavior is created by putting states, each of which represents a specific atomic action, in order according to a reasonable sequence, so that they represent the behaviour captured in state charts. In this respect, behaviors undertake an important role in sharing domain semantic. In addition, because of the logic programming paradigm, capabilities of agent-hood modeling characteristics, continuous/discrete simulation support and symbolic time management, AdSiF provides a perdurantist modeling environment. Perdurantist approaches assume that objects have four dimensions (spatial and temporal) and only briefly exist during the different stages of their life span [9]. That is entities only exist for a period of time and continually change over such periods. Such entities are unfolding themselves over time in successive temporal parts [10]. Therefore, objects are viewed from the past, present and future. According to this paradigm, entities are usually referred to as 'space-time worms' or a slice of such a worm [11] given that they are identified based on space and time dimensions.

The paper focuses on ontology-based modeling enriched with logic programming and reasoning. It aims to give an answer and a means of connecting logic and ontology with each other. The role of relations, behavioral aspects and reasoning mechanism are also emphasized in ontology-based modeling and the perdurantist modeling approach is extended by a reasoning mechanism. The extension allows modelers to manage behaviors depending on reasoning results and also gives well-designed support for time-delayed systems.

One of the main aims of the study is to match the predicate logic ontological commitment with the ontological commitment presented here and bring them into a comment framework to handle behavioral management in simulation. A language support is provided for this purpose, beginning with an ontological commitment and covering design and coding phases, including conceptual model description.

The paper is organized as follows: The first section introduces brief information about ontology-based modeling. The second section (2) focuses on logic programming and how it supports entity description and a relation concept between entities. In the fourth section, AdSiF ontology support is presented in the following section with the paradigms it covers.

1 Ontological Commitment of AdSiF

An ontological commitment refers to a relation between a language and certain objects postulated to be extant by that language. The overall philosophical project of ontology is categorized into at least two parts: the first part is about what there is, what exists and what the thing is made from and the second part concerns what are the most general features and relations of these things.

Concepts, relations of the phenomenon and the objects surrounded by this phenomenon are strongly related to the conceptual model and the conceptual model paradigm that is most effective for ontology modeling. From this point of view, not only does AdSiF provide multi-modeling paradigm support, but also it combines the paradigms into a single paradigm termed state-oriented paradigm. This gives a rich expressiveness that is one of the quality metrics of an ontology [12]. The paradigms, which are supported by AdSiF and combined in state-oriented programming, are logic programming, aspect-oriented programming, agent-based programming and object- oriented programming.

A system has a time base, inputs (events), states, behaviors, a reasoning mechanism and a mechanism that manages the dynamic characteristics of the system. In AdSiF, the dynamic characteristic of the system is represented by the behavior descriptions (in the specialized state charts). As a framework, AdSiF promotes a set of design rules and presumes a design skeleton, which is based on its programming paradigm, called the SOP paradigm, and its ontological view. AdSiF provides an ontological view, which is defined as follows in terms of the paradigms, on which AdSiF are based.

AdSiF's ontological commitment covers time, space and provides an answer as to what exists, posits a type of relation definition between existances and it is given below.

Entities live in a certain environment and have their own properties that distinguish them from each other and an atomic action that manages their properties (OOP perspective). The atomic action creates the interactions that change the environment where the entities reside and share the interactions with other entities as a communication element. The entities sequentially implement their atomic actions in a reasonable semantics called behavior, and the behaviors are executed in parallel and/or sequentially.

Each atomic action is wrapped by a state that constructs the behaviors (SOP perspective). The entities interact with one another using event transactions and constitute relations among one another. An interaction has autonomy, reactivity, and a goal — (Aget Oriented Programming - AgOP perspective). From a taxonomic view, the behavior categories of an entity represent different behavioral aspects of the entity — (Aspect Oriented Programming- AOP perspective).

An entity has beliefs and facts about the environment and about the other entities with which they share the environment. These beliefs and facts constitute a fact dual world envision that contains the entity; the envision may have a set of goals to succeed and a reasoning mechanism with a set of decision-making algorithms — Logic Programming (LP perspective).

AdSiF's ontology covers the common properties of entities following an inheritancy path. Common properties are defined as public or protected and inherited from base models, as in OOP. Agenthood perspective also gives another property of the ontology, such as being in an environment, reacting to events happening around and trying to achieve certain goals by behaving proactively. Also, an entity changes its behavioral aspects depending on the conditions which it is in .

2 Logical Envisonment and Behavior Management by Reasoning

As presented in the ontological commitment, simulation models and agents have facts (beliefs) about the environment in which they exist and truth-preserved predicates to infer new facts, relations and identities. The time-stamped facts about simulation or agent environment not only constitute a fact dual world representation as an inner representation of the environment in which the models are, but also the dual world retains knowledge of the time axis with past values. Dual world representation is defined as an inner representation of the environment created by the entity.

Each entity has its own representation of the environment in which it is in and this is constituted via sensed and inferred information. This allows modelers to associate truth level and define temporally valid (for a certain duration) knowledge for both truths about simulation models sharing the same environments and the relations (dependencies) between them.

This can be seen as a modeling characteristic supporting the *rigidity* notion [12] of ontology-based modeling. The capabilities are enriched by a logic paradigm which turns a perdurandist model into a model which has cognitive capability and can manage relations dynamically.

Semantic representation of an agent or a simulation model (namely, an entity) is represented by behavioral descriptions. Any interaction, in other words any event transition between simulation entities and agents, causes a set of behavioral reactions, either reactive or proactive. Each behavior taken consists of a series of actions connected with each other logically. In addition to the behavior that is activated by an event, activated or cancelled by a condition, generally, managed behavior after a series of reasoning processes is a good example of shared semantic. An example for activating a behavior as a result of reasoning is given in Section 5.

Meta-knowledge, higher-order rules are vital for both agents and ontology-based modelling. To enable the development of high-level easy-to-configure agent behavior, it is important to provide agents with the means to reason about their surrounding environment using a generic reasoning mechanism and knowledge base. The agents must be able to analyze unexpected situations to dynamically adapt their behavior to achieve their personal goals [13]. This means agents must be capable of evaluating situations using abstract, higher-order rules.

One of the higher-order sources comes from the answers to such ontology questions as "what is it?" or "what can be said to exist", because the answers given consist of a set of relations from a specific instance to more abstract ones. In other words, an answer becomes more and more generalized up to the infinitesimal. The answers constitute a fact set for rules binding to the behaviors. This provides a means to be able to make a decision about any new instance inserted into the knowledge base generalizing it.

In AdSif, a rule consists of a head and a body, similar to Prolog syntax. The head and body are connected by a symbol :-, which is made up of a colon : and a hyphen –[14]. The ":-" is pronounced *if*. Return parameters of a rule are used to bind to Boolean logical expressions (**Figure 1**) and rule truth value, which shows whether it has succeeded or not, is used as bool value. Parameter-binding pseudo code is shown in **Figure 1**.

In the figure, the parameter numbered with *no* of the rule named *rulename* is compared with a value *val*. Boolexpreation *name0* can be used as a trigger (activation) condition, cancel condition, suspend condition or a resume condition for a behavior, a guard constraint (whether the state is activated or not) for a state, temporal relation, or a sending condition for an event, etc.

```
rulename(param0,param1…paramN):-rule0(paramset),
rule1(paramset),….
   Boolexpname=<″name0″>
   Type=″<comparison/logical>″ opera-
tor=<″EQ,GT,LT,GE,LE/and,or″>
   <leftvalue type=″predicate″ predicate-
name=″rulename″ OutputFieldNo=″no″>
   <rightvalue  type=<″constant/function/etc..″>
value=val
```

Figure 1: Rule Representation.

An agent can behave in different ways depending on the situation it is in based on its dual world representation (inner model retained in the knowledge base). In this sense, behaviors can be categorized according to well-defined world views and each category that describes how the model behaves under certain conditions. The condition is defined as a decision model and is used to activate or deactivate the related category or categories [15]. It is a very useful property to be able to change a model world view in both design time and run time. It can be seen as a dynamic description. The parameters and truth value of the rule are also used to shift from one aspect to another. It is shown in **Figure 2** as pseudo code. The main point is to make a decision based on an ontological description of entities and manage behaviors based on the decision.

3 Ontology-based Modeling

Ontology is a term that originated in philosophy and refers to the systematic explanation and study of the nature of existence, or being [16]. Ontologies are composed of concepts or entities, relations between these concepts (or entities) and axioms to limit the interpretation of concepts for a real world phenomenon.

Whereas in computer science, ontologies are recognized as a useful means for achieving semantic interoperability between different systems and are key enablers for sharing precise and machine-understandable semantics among different applications and parties [10]. In the simulation world, it is aimed to use ontology to give meaning to entities at different abstraction levels by binding rules (axioms) and defining behaviors. Similar to the ontology definition used in information systems, in the simulation world a common language is also developed to be used for an entity at each level. In modelling and simulation, the use, benefits and the development requirements of Web-accessible ontologies for discrete-event simulation are investigated [17].

```
<BehaviorList ListA>
     <behavior A>
     <behavior B>
     <behavior C>
     ..
<driveCond>
   <activation cond=″<gettingLower>″
   <cancel Cond=″<>″>
</driveCond
</ BehaviorList >
```

Figure 2: Shifting Aspects by Reasoning.

Ontology studies are related to questions such as "*what is it?*" or "*what exists*" and "*what relations has it?*". Looking for an answer for the first question is strictly related to generalizing a particular situation and/or an entity. This allows us to create more general rules and management capabilities for behaviors using higher-order reasoning. The second question is related to the relation concept. An entity (both an agent and a simulation model) may have relations with other objects.

In AdSiF representation, the relations are categorized under three main headings: 1) Predicate (Logical), 2) Functional (Behavioral) and 3) Structural. A predicate relation consists of facts (e.g. position information of a missile time-by-time), higher-order descriptions (e.g. missile types and their behavioral patterns, such as being in boost phase, etc.) and rules (decisions about what an object is or how to behave).

Answers to "*what is it?*" are given at different abstraction levels each time it is asked. The answer given on each occasion makes the entity definition one more level abstracted. For example, the answer "*it is a fighter*" given for a question asked about an F16. The F16 is a multi-role fighter aircraft designed in the 70s and still being produced and actively used by the airforces of various nations. After having the answer, a new question arises, such as about "*what a fighter is*". The answer carries the definition one more level up, such as *it is a plane*, and so on.

This continues until arriving at the most primitive, minutest definition, such as *it is an object*. All these abstracted declarations are linked to each other with a predicate such as

is_a_kindof (f16, fighter), is_a_kindof(fighter, plane)

and continues until arriving at an answer referring to the most basic well known entity or concept. Quality of the answer is measured by whether a behavior set and a set of properties are defined at each level, determined by the answer given or not.

As mentioned earlier, the relation concept is considered to be functional, predicate or structural. Functional relations that are established between two parties force a definite behavior set for both sides. In this sense, the relations define behavioral templates at different abstract levels. The relations create a behavior set which is activated for the entities on both sides of the relation. The behavior set being activated differs depending on the model type, entity abstraction level and environment or entity state vector.

The entity drives a function in regard to any other entity; for example, "*f16 carries missiles*" shows a functional relation and the relation triggers functions or behaviors on both of the side objects, namely *f16* and *missiles*, at the phases in which the relation is established and detached. In **Figure 3**, the behavior sets are shown to be executed in the "*carry*" relation establishment phase and detachment phase for both F16 and missiles. F16 executes "*RelationBehaviorList.A*" and "*RelationBehaviorList.B*" any time relation is established and detached, respectively. Similarly, missiles does the same thing in the same phases for "*RelationBehaviorList.C*" and "*RelationBehaviorList.D*". The representation presented in **Figure 3** is executable and is interpreted by AdSiF core engine.

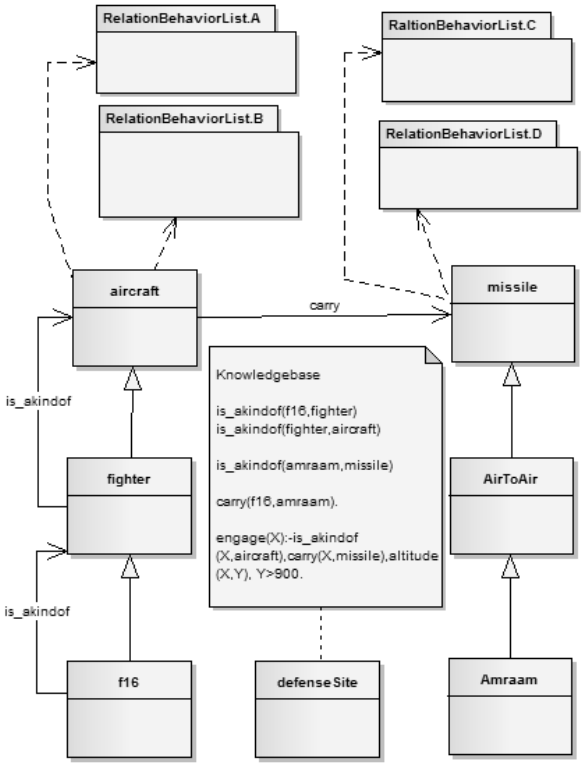


Figure 3: Functional Relation Declaration.

A predicate relation defines declarations between two objects or among more than two objects constituted on stative descriptions of the objects. The declarations are taken into consideration as rules that are used to manage a behavior or a set of behaviors. As an example, let us take the premises "*F16 is equipped with missiles"* and "*F16 is over 1000 ft at time t0 "* into consideration.

They can be symbolized as *equipedWith(f16,missile)* and *altitudeOver(1000, t0).* In the first, relation is representation and we have two objects, F16 and a missile or a set of missiles, and the second is related to the object itself. Whereas the functional relation is categorized as a relation that forces objects on both sides of the relation, a specific behavior set, and the predicate relation is also related with third object decision. The object that knows or infers the relation between two objects uses the information for decision-making and the result of the decision triggers a behavior or a set of behavior. An entity possibly uses relations that it has to handle its own behavior. In other words, an entity can manage its own behaviors using its own predicate relations.

For example, F16 (or a missile) that has predicate *equippedWith(f16, missile)* can trigger a behavior set by using the predicate.

Similarly, any other model, such as an air defense site, can detect the relation between F16 and missiles and uses this knowledge as a rule to trigger engagement behavior. The inference made by the air defense site is translated into natural language, such as "*I (the* air defense site*) have detected an f16 with missiles*". The inference is evaluated using a rule such as "*if an aircraft is detected with missiles and its altitude is over 900 ft, then start an engagement with it*" (**Figure 3**). It is clear that this is triggering a behavior.

In the rule given here, it does not say directly anything about F16 specifically, but it was previously known, after a set of "*what it is*" questions, that an F16 is an aircraft. In this sense, predicate relation is strongly related to propositional logic. The predicates detected or inferred about entities in the environment are kept as time-labeled facts in a knowledge base (such as position information set time-by-time). The knowledge base constituted by the facts is defined as a dual world representation of the environment and also consists of information other than relation predicates.

It is important that ontologies are of a good quality, in order that they serve their intended purposes and be shared as well as reused by different applications [12]. A good quality depends on how clear the answers are to the "*what it is*" question, constituted of depth of model abstraction sequence (such as F16 is a kind of plane, plane is a kind of aircraft, etc.), relations defined between entities and also the rules and behaviors attached to both the relations and abstract model definitions. Relations follow an inheritance path. A functional relation constituted for any specific entity is valid for the entity derived from it, in other words, its child. This is valid for predicate relations, but exceptions can be made. In this respect, we can say the *carry* relation (F16 carries missiles) is valid for a fighter derived from the F16 model. The generalized form of that relation is "*An aircraft carries missiles*". Similarly, the relation is also valid for any type of missiles that are derived from the missile.

The relation mentioned here is constituted in run time. In design time, composition and aggregation relations are defined. In simulation execution, the entity manages simulation components that it aggregates and/or composes. Management consists of event handling and time management. Composition and aggregation are defined very similar to that between classes [18].

Composition and aggregation are a way to combine simple objects and data types, in this case, entities, into more complex ones. If the more complex one is destroyed, the simple object is also destroyed. Because, it is a non-detachable part of the owner (more complex) entity. But in aggregation cases, they can still survive without the owner entity. One can imagine the simple entity as an attribute of the complex one.

Another difference between composition and aggregation is seen in task sharing in the model design of AdSiF. The entity undertakes the event handling and time management tasks of its components (both composed and aggregated). In **Figure 4**, dashed and solid arcs show aggregation and composition relations, respectively. EntityS, EnitityA and EntityB coordinate with each other and handle their own event handling and time management. But Time requirement of EntitiyB depends on EntityZ requirement and EntityA time requirement depends on the entity to which it is connected. EntityZ time requirement also depends on the entities to which it is connected. The only interface of aggregated and composed entities is their owner entity, and, as a result, they distribute and collect their event messages across owner entities. The behaviors are still separated and the owner entity never intervenes in the behavior of its components.

As seen in **Figure 5**, F16 undertakes behavior and time management of external fuel tanks and a control panel. This means an entity is capable of managing the simulation loop of sub-entities. In **Figure 5**, F16 has two types of behavior list. The behavior list named "*f16 behavior list*" consists of the F16's or, more generally a fighter's, capabilities. The behavior list named "*Composed & Aggregated model management behavior list*" is inherited from the base model and the behaviors allow it to create time and event managements of subcomponents.

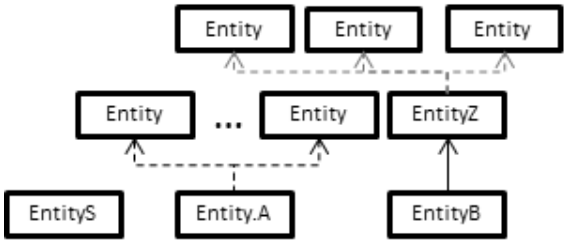


Figure 4: Sub-Simulation Loops.

From an ontology point of view, an entity may have several atomic sub-components, each of which is an individual entity. They maintain their own behaviors and data structures, such as attributes, custom data, etc. Time requirement and delivering events of sub-components are managed by the owner entity (in the example, it is F16) and management behaviors are inherited from the base model that is an extended AdSiF entity class. This gives a new dimension to AdSiF's ontology. An entity (or an existence) may be formed by a collection of single entities. Nevertheless, the owner entity has interfaces with the environment.

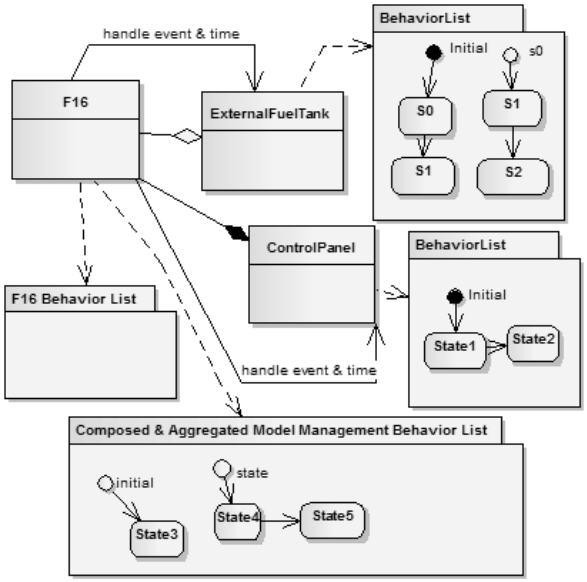


Figure 5: Composition and Aggregation.

4 Ontology-based Modeling Language Support

Fundamental ontological notions are summarized by [19] as identity, a unity that is related to the problem of distinguishing the parts of an instance from the rest of the world by means of a unifying relation that binds them together (not involving anything else); rigidity, which is a property it necessarily holds for all its instances; and dependence, which is the property of an entity that is dependent of the existance of other entities.

The ontological commitment of AdSiF embraces the identity notion by giving a unique id to each object; the unity notion by relations, the rigidity notion by time-framed and consistent facts about objects, and the dependence notion by composition and aggregation depending on design choices; it also extends notions by logical commitments, behavioral semantics and conditional aspects notions.

In the literature, it is possible to find several ontology-based modeling tools, such as Anemone [20], Protégé [21], OilEd [22], Apollo [23], OntoLingua [24], OntoEdit [25], WebODE [26], Kaon [27], DOE (Differential Ontology Editor) [28], WebOnto [29] and K-Infinity [30]. Anemone provides a methodology which differs from previous methodologies in the way that it defines concrete development steps, to facilitate use by both novice and expert ontology developers. This methodology is also supported by ontology design patterns and a prototypical ontology development tool [20]. System Entity Structure and Model Base (SES/MB) is developed for modeling and simulation domain [31].

The distinguished and prominent supports provided by AdSiF are brought by the logic programming paradigm and relation concept. Beyond allowing modelers to define environments, objects and relations between objects at meta-level, it also affords the possibilities of developing a reasoning mechanism simultaneously operating on past and present time-stamped knowledge - such as

```
position(f16, x0,y0,z0, time0), position(f16, x1,
y1 ,z1, time1), position(f16, x2 ,y2 ,z2, time2),
```

.etc.) and driving behaviors depending on inferences - such as

```
gettingLower(X):-position(X,_,_,Z,T),
position(X,_,_,Z2,T2), T2>T, Z2>Z
an object X getting lower)
```

which constitute future information sets. Driving a behavior is defined as activating, cancelling, suspending and resuming it. This is seen as distilled knowledge inferred from a set of meta-level knowledge and declarations that are placed at the kernel of ontology-based modeling.

Both relation predicates and facts about the world maintained in the knowledge base are used for decision. Decision-making results in a truth value and a set of output parameters. Truth value is directly bound as a drive condition (in **Figure 6**, drive conditions are shown in pseudo code form) to behaviors. The output parameters are used as input parameter for boolean expressions, such as logical expressions or comparisons. The boolean expressions are also used to manage behaviors, such as to activate, cancel, suspend and reactivate and as guard constraints in any place required.

First order predicate logic (FOL) and propositional logic are the fundamental reasoning techniques used in predicate relation and fact-based reasoning. Both logics are perfectly matched with AdSiF's ontology. FOL consists of variables for individual objects, quantifiers, symbols for functions and symbols for relations [32].

```
<behavior A>
     <state Set>
<driveCond>
   <activation type="predicate"
cond="<gettingLower>"
   <cancel type="boolExp" Cond="name0">
   <suspend Cond="<>">
   <reactivate Cond="<>">
</driveCond
</behavior>
```

Figure 6: Drive Conditions.

5 Ontology Example

A simple example from a defense modeling and simulation is chosen to show how the concepts are implemented. In **Figure 7**, the relations between objects are depicted. The relation between different objects with the same name activates different behavior and action sets. Each relation states a meaning depending on the behavior space of models that are the relation constituted between them. This is also valid for the abstraction level of models. The behavioral description of the relations is shown in **Figure 8** for the relation *use* between "*F16*" and "*Missile*".

The relation rule is applicable to all types of F16 and missiles and any type of models derived from these models. In the plane domain of **Figure 7**, it can be seen what behaviors are executed in both activations, which constitute the relation and passivation phase, which means breaking it. All ontology commitment is not given here; a good example of missile phase and dynamic modeling can be seen in [33].

In Figure 9, an ontological description is given from the more abstract class level up to instance level. The second part of the figure shows a set of predicates inferring what the target is and what kind of missile is to be fired using the facts given in the third part and populated by sensory data. The sensors providing detection information send their detections to commanders, which drive F16, using the relation "informs". The decision triggers the behavior seen in Figure 10.

The behavior selects a missile as a result of an inference using target information received from event-named detection and a set of predicates, facts and ontological descriptions given in the knowledge base shown in Figure 9.

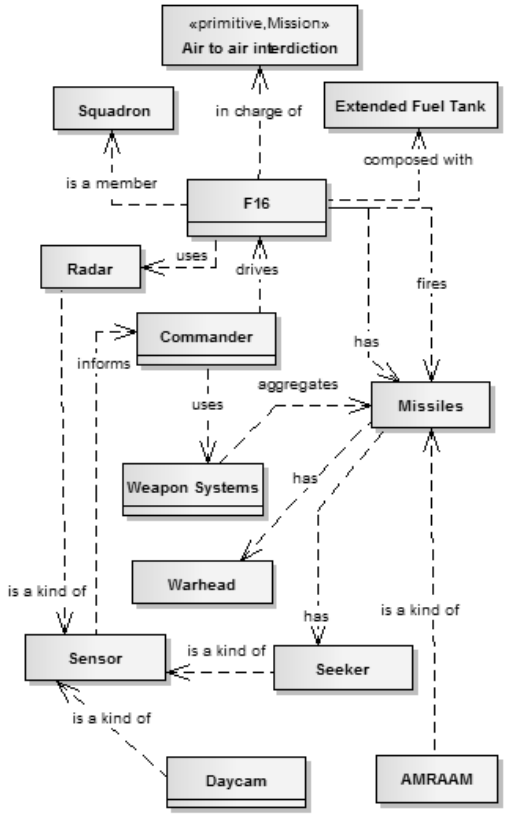


Figure 7: Relations Between Objects.

6 Discussions

AdSiF programming approach and state-oriented paradigm, which is the main programming paradigm, expands a perdurantist modeling approach to a reasoning-capable model by a logic programming paradigm and ontology-based modeling world view.

The ontological commitment that AdSiF is based on matches the predicate logic ontological commitment and bring them into a comment framework to handle behavioral management in simulation.

Logic programming and modeling paradigms enrich ontologies by giving inference capability, keeping the information as time-framed and allowing it to expire, defining relations between objects and conditions in them.

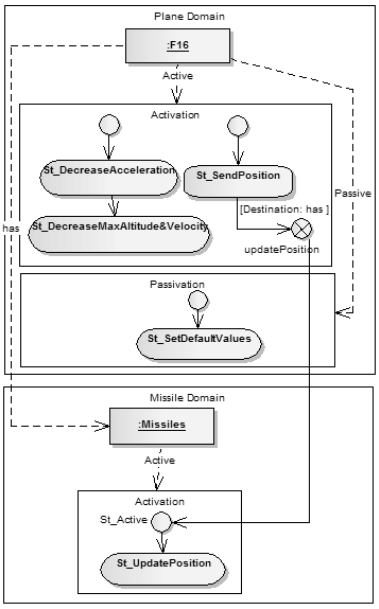


Figure 8: Behavioral Descriptions

```
thing('F16', 'Fighter')
thing('Fighter', 'Plane')
thing('Plane','Aircraft')
thing('Aircraft','Object')
thing('Missile','Weapon')
thing('AMRAAM','Missile')
thing('Weapon','Object')

decision(Target, 'Plane', 'Missile'):-detectionRadar(Target, Position.Altitude, Vel,Direction,T0),
                    detectionDaycam(Target, DayCam, Width, Length, T1),
                    Vel>=200, 800>=Vel, Position.Altitude >=50, Width>=3,
                    10>=Width, Length>=4, 15>=Length.
decision(Target, 'F16','AMRAAM'):-detectionRadar(Target, Position.Altitude, Vel,Direction,T0),
                    detectionDaycam(Target, DayCam, Width, Length, T1),
                    Vel>=500, 1500>=Vel, Position.Altitude >=50, Width>=3,
                    9>=Width, Length>=5, 12>=Length.

.... more rules ....

Fact Set

detectionRadar(TargetNumber, SPosition,Velocity,Direction,Time)
detectionDaycam(TargetNumber, Width, Length , Time)
```

Figure 9: A Part of the Knowledge Base

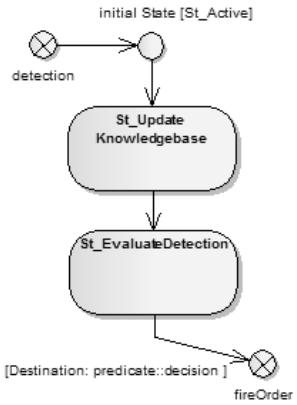


Figure 10: Driving Behavior by Inferences

In addition to inference, logic programming combined with state-oriented programming allows modelers to model domain information at meta-level. Each meta-level system model has a model family rather than a specific implementation model.

This study points out how to use concepts and axioms defined as logical premises and how the relations between higher-order entity descriptions are used to combine in a behavior management structure. Logical premises define model structures at different abstraction levels of domain information. Relations provide indirect interaction based on entity description, not direct interaction with objects, and also straightforward behavioral descriptions. Similarly, conditional aspect management, that means, shifting from one aspect to another presents a dynamic ontology in run time.

The proposed solution also provides a logic based solution for time-delayed systems [34], allowing simulation and agent models to use time-stamped facts stored in the knowledge base of the model. A time-delayed system needs an earlier value of the decision variables and the time-labeled facts in the knowledge base of entities provide a good solution to the problem.

References

[1] M. F. Hocaoglu, "AdSiF : Agent Driven Simulation Framework," *Hunstv. Simul. Conf. - HSC2005*, 2005.

[2] M. F. Hocaoğlu, "AdSiF : Agent driven Simulation Framework," in *USMOS 2011- National Defense Application and Modeling & Simulation Conference -(in Turkish)*, 2011.

[3] M. W. R. H. Bordini, J. F. Hübner, *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley Series in Agent Technology, 2007.

[4] R. Kay, "Aspect-Oriented Programming," *Computerworld*, vol. 37, no. 40, 2003.

[5] F. Pfenning, "Logic Programming." Carnegie Mellon University - Lecture Notes, 2007.

[6] M. F. Hocaoğlu, "AdSiF: Developer Guide." Agena Information & Defense System Ltd. www.agenabst.com, Istanbul, 2013.

[7] D. Harel, "State Charts: A visual formalism for Complex Systems," *Sci. Comput. Program.*, vol. 8, pp. 231–274, 1987.

[8] Y. Shoham, "Agent-Oriented Programming." Technical Report STAN-CS-90-1335- Stanford University, 1990.

[9] T. A. Hales, S.D., Johnson, "Endurantism, Perdurantism and Special Relativity," *Philos. Q.*, vol. 53, no. 213, pp. 524–539, 2003.

[10] M. M. Al-Debei, M. M. Al Asswad, S. de Cesar, and M. Lycett, "Conceptual Modelling and the Quality of Ontologies : Endurantism Vs Perdurantism," *Int. J. Database Manag. Syst.*, vol. 4, no. 3, pp. 1–19, Jun. 2012.

[11] T. Sider, *Four Dimensionalism: An Ontology of Persistence and Time*. Oxford University Press, 2001.

[12] N. Guarino, D. Oberle, and S. Staab, "What is an Ontology ?," S. Staab and R. Studer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–17.

[13] F. Béhé, S. Galland, N. Gaud, C. Nicolle, and A. Koukam, "An ontology-based metamodel for multiagent-based simulations," *Simul. Model. Pract. Theory*, vol. 40, pp. 64–85, Jan. 2014.

[14] C. S. M. W.F. Clocksin, *Programming in Prolog*, Second. Cambridge, England: Springer-Verlag Berlin Heidelberg New York Tokyo, 1984.

[15] M. F. Hocaoğlu, "Aspect Oriented Programming Perspective in Agent Programming," in *USMOS 2013- National Defense Application and Modeling & Simulation Conference -(in Turkish)*, 2013.

[16] A. G. Bruzzone, R. Mosca, R. Revetria, E. Bocca, and E. Briano, "Agent Directed HLA Simulation for Complex Supply Chain Modeling," *Simulation*, vol. 81, no. 9, pp. 647–655, 2005.

[17] A. P. Miller, John A., Baramidze, Gregory T., Sheth and P. A. Fishwick, "Investigating Ontologies for Simulation Modeling," in *ANSS '04 Proceedings of the 37th annual symposium on Simulation*, 2004, p. 55.

[18] B. Stroustrup, *The C++ Programming Language*, 4th Editio. Pearson Education Inc., 2013.

[19] N. Guarino and C. Welty, "Towards a methodology for ontology based model engineering."

[20] T. Özacar, Ö. Öztürk, and M. O. Ünalır, "ANEMONE: An environment for modular ontology development," *Data Knowl. Eng.*, vol. 70, no. 6, pp. 504–526, Jun. 2011.

[21] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe, S. Jupp, G. Moulton, N. Drummond, and S. Brandt, "A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools," 2011.

[22] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens, "OilEd: A Reason-able Ontology Editor for the Semantic Web," *Lect. Notes Comput. Sci.*, vol. 2174, pp. 396–408, 2001.

[23] M. Kamil Matoušek, LubošKrál, "Apollo CH Manual," no. August 2004. Knowledge Media Institute, The Open University, UK, pp. 1–23, 2004.

[24] J. Fikes, R., Farquhar, A., Rice, "Tools for assem- bling modular ontologies in Ontolingua," in *AAAI/IAAI*, 1997.

[25] S. Sure, Y., Angele, J., Staab, "OntoEdit: Multifac- eted inferencing for ontology engineering.," *Data Semant.*, vol. 1, pp. 128–152, 2003.

[26] G.- Arpirez, J., Corcho, O., Fernandez-Lopez, M. and A. Perez, "WebODE: A scalable workbench for ontological engineering," in *First International Confer- ence on Knowledge Capture (KCAP'01)*, 2001, pp. 6–13.

[27] B. Volz, R., Oberle, D., Staab, S., Motik, "KAON SERVER: A semantic Web management system," in *Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003*, 2003, pp. 20–24.

[28] V. Isaac, A., Troncy, R., Malais, "Using XSLT for interoperability: DOE and the travelling domain experi- ment," in *Proceedings of the Second International Work- shop on Evaluation of Ontology-based Tools (EON 2003), Sanibel Island, FL.*, 2003.

[29] O. Domingue, J., Motta, E., Corcho Garcia, "Knowledge modelling in WebOnto and OCML: A user guide." http://new.euromise.org/mgt/webonto/main.html - Knowledge Media Institute, The Open University, 1999.

[30] S. Youn and D. Mcleod, "Ontology Development Tools for Ontology- Based Knowledge Management Ontology Development Tools for." http://research.create.usc.edu/cgi/viewcontent.cgi?article=1104&context=nonpublished_reports - CREATE Research Archive, 2006.

[31] P. E. H. B. P. Zeigler, *Modeling & Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange*. Elsevier, 2007.

[32] P. N. S. Russel, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[33] U. Durak, H. Oğuztüzün, and S. K. İDER, "Ontology-Based Domain Engineering for Trajectory Simulation Reuse," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 19, no. 08, pp. 1109–1129, Dec. 2009.

[34] B. P. Zeigler, *Theory of Modelling and Simulation*. Robert E. KRIEGER Publishing Company Malabar, Florida, 1976.

A General Concept for Description of Production Plants with a Concept of Cubes

Niki Popper[1,2 *], I. Hafner[1], M. Rössler[1], F. Preyser[2], B. Heinzl[3], P. Smolek[4], I. Leobner[4]

[1] dwh Simulation Services, Neustiftgasse 57-59, 1070 Vienna, Austria; **niki.popper@dwh.at*

[2] Institute of Analysis and Scientific Computing, Vienna University of Technology, Wiedner Haupstraße 8-10, 1040 Vienna, Austria

[3] Institute of Computer Aided Automation, Vienna University of Technology, Treitlstr. 1-3, 1040 Vienna, Austria

[4] Institute for Energy Systems and Thermodynamics, Vienna University of Technology, Getreidemarkt 9, 1060 Vienna

Simulation Notes Europe SNE 24(2), 2014, 105 - 114
DOI: 10.11128/sne.24.tn.10247
Submitted: Sept. 15, 2014 (selected ASIM SST Post-Conf. Publ.);
Revised: Oct. 25, 2014; Accepted: October 30, 2014;

Abstract. The goal of the project BaMa (Balanced Manufactoring) is to develop a simulation-based method for monitoring, predicting and optimizing energy and resource demands of manufacturing companies. Considering the economic success factors time and costs, a new modelling and simulation concept will be integrated in the research project to implement an energy and cost foot printing. A modular approach that segments a production facility into "cubes" will be developed. Cubes have a clearly defined interface and represent a certain physical behaviour that contributes to the energy balance of the overall system. This article shows the basic concept how cubes are defined and how formal concepts for interfaces, system behaviour, and hierarchical layout are described.

Introduction

Balanced Manufacturing (BaMa, the project is running from 2014 until 2018) will develop a simulation-based tool for monitoring, predicting and optimizing energy and resource demands of manufacturing companies under consideration of the economic success factors time, costs and quality. Goal of the modelling approach - which is done in the first part of the project –should be the development of methods, which are able to integrate all building blocks of the facility (production, building, energy, logistics, management system) with one approach. This phase of BaMa started with a thorough system analysis and the definition of the methodology. In order to address these challenges, systematic approaches, as described by Thiede et al in "A systematic method for increasing the energy and resource efficiency in manufacturing companies" [1] have been analysed. A modular approach was chosen, that segments a production facility into so called "cubes". In the first step the features of the cubes were defined. Cubes have in addition clearly defined interfaces and represent a certain physical behaviour that contributes to the energy balance of the overall system. Nevertheless all cubes should be built up with the same architecture.

One of the main goals of BaMa is to monitor and compute energy and resources consumption. For doing so, based on the cube related energy and resource flow analysis, the method should be able to generate a specific product-footprint for every product running through the "cube system". The product footprint represents a products expenditures concerning cost, time, energy and the environmental impact such as resulting carbon emissions in the product life cycle phase within the factory.

Of course there are already comprehensive planning tools, such as [2], which also have been analysed. Regarding this analysis BaMa will also be implemented inside a customised toolchain. The toolchain (Balanced Manufacturing Control, BaMaC) allows energy efficient operation, design and refurbishment of production plants under competitive conditions, with regard to minimal energy and resource consumption. Tools to assist energy conscious steering of a plant during operation will be developed as suggested by K. Bunse et al in [3]. BaMaC will contain three core modules*:*

The modules in detail will be able to support the three tasks: *Monitoring:* data on resources consumption will be aggregated and visualised, data can be implemented into simulation of cubes.

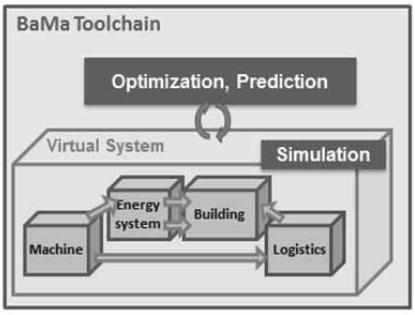


Figure 1. Future Modules of the BaMa Toolchain. The simulation approach has to fulfil various demands.

Prediction: allows forecasting of overall energy demand of the plant based on the product-footprint and the production schedule. *Optimisation:* based on data and numerical simulation-models of the cubes, this part of the tool chain will improve the plant operation with regard to the optimisation targets energy, time, costs and quality.

By integrating the four main optimisation fields building, energy system, production, and logistics equipment BaMa will be applicable to a variety of industrial sectors. It will serve as a basis for a software tool chain which will be integrated into industrial automation systems, such as ERP or MES. The toolchain will introduce energy efficiency as a steering parameter into the control centre, thus enabling manufacturing companies to balance energy efficiency and competitiveness in their continuous operation strategies.

To satisfy the described demands of BaMA and BaMaC the cube concept needs to fulfil a variety of characteristics. The concept has to fit a variety of applications i.e. it should be able to integrate all relevant building blocks of the facility (machines, energy system, logistics, …) with the same architecture. It is used as formal description of the real production plant and also as basis for models of the system. This modelling should be possible more or less "directly", without much amount of work for translating. The cubes must have clearly defined features and interfaces and the system should be able to generate a specific product-footprint for every product running through the "cube system". And finally of course implementation should be possible easy, fast and stable.

1 Motivation of BaMa - Footprinting

One of the most interesting demands – and main goal - in BaMa is the implementation of a comprehensive foot printing for industrial production plants. Industrial production accounts for 40% of the energy consumption of Europe, with an estimated potential for reduction of 30% to 65% [4]. A common top-down approach to identify the environmental impact of products is to assess the Carbon Footprint of Products (CFP) on a one-year-basis. This procedure is important for raising awareness. However, for the purpose of optimizing plant operation it is not well suited, because the results can vary on a large scale due to the lack of transparency of different methods [5], missing standardisation [6] and the lack of reliable data [7]. In addition the CFP fails to incorporate the diversity of different types of expenditure that go into the manufacturing of products.

In order to address these issues the BaMa bottom-up approach for aggregating a product footprint during the production phase of the product life cycle was proposed. This method allows for real-time evaluation of a batch or even single product using monitoring or simulation data. The definition of a significant footprint sets product success factors in context with its ecological impact. In particular energy, costs, carbon emission and time will be captured and visualised for the transformation process a product undergoes within the plant. Each part of the plant contributes to the product's energy, cost or time consumption, as well as carbon emission, which accumulates the product footprint. The energy used by production machines, auxiliary infrastructure, logistics and the building is aggregated from the entry of the raw materials to the departure of the finished good. The integral footprint of all products produced in a year match the yearly carbon footprint of the plant exactly. So comparability with conventional studies is achieved.

From this bottom-up approach different challenges arise. For example, the incorporation of standby-, setup- and ramp-up times, the energy consumption of the administration and the allocation of different products and by-products manufactured at a machine are some of the problems. The necessity to calculate mean values and dividing them between different products demands for a way to assess the degree of which each product is responsible for the generated footprint. One can easily see that measurement of data for this applications and modelling of such processes is challenging. Implementation would strongly benefit of a clear defined modelling concept and approved, straight forward methods. The cube approach, in which the system is described through black boxes (cubes) connected through inputs and outputs has to manage to map the complexity of a manufacturing facility in the necessary detail and breaking down the plant into its elements.

The inputs and outputs of cubes can be material, energy or information flows. Energy flows carry a qualifier to determine the different expenditures, including carbon emission and monetary value. The products in the material flow accumulate the footprint by aggregating the cost, energy consumption, carbon emission and time inside the system boundary.

2 Requirements for Cubes

Based on the previous findings, a methodology for conducting a comprehensive system analysis of a production plant in preparation for the implementation of Balanced Manufacturing had to be developed. The methodology should be formulated at a generic level to ensure its usability in a variety of production facilities.

As described the basic element of this system analysis consists of the so called cubes. The idea was that cubes constitute subparts of a system "production-plant" and have the following properties:

- defined boundaries,
- interfaces to other cubes,
- a certain physical behaviour that contributes to the energy balance of the system
- and usually some degree of freedom to be influenced for optimisation.

To put it differently, the boundaries of sub systems in terms of energy-, material- and information flows had to be thoroughly defined to intersect the whole system into observable parts. The characteristics and attributes of cubes should be specified in a generic way in order to guarantee the applicability for all parts of the plant and for different kinds of productions. A cube could be a machine tool, a chiller, a baking oven, the production hall or a utility system. The definition of the cubes should allow implementing the described product-footprint evaluation, which sets the product success factors in context with its ecological footprint.

In particular the resources energy, costs and time will be captured and visualised for the transformation process a product undergoes within the plant. Each cube should contribute to the product's energy, cost or time consumption within the production plant which accumulates the product footprint. The product-footprint should be made up of a high number of originally independent data streams that are aggregated in a time-synchronised manner. So also methods for suitable data aggregation and fragmentation should be found and described.

So our approach leads us to the following process. (see Figure 2) . In the first step we analyse general systems of production plants. As a matter of fact in BaMa a number of basic applications of real world system were taken to be analysed (e.g. production facilities of semiconductors, bakeries, metal processing industries, …). Based on these approaches several specific cubes are defined with a variety of needed features for input, output, system behaviour, system variables, changing processes and many more. An additional general analysis is done and a generic cube definition is formulated. This cube definition is one step before the formalisation of the modelling concept we will introduce. The modelling concept (formal model) will especially need to be able to handle continuous and discrete processes running through the "cube system". The last step is the implementation of simulation applications for BaMaC.

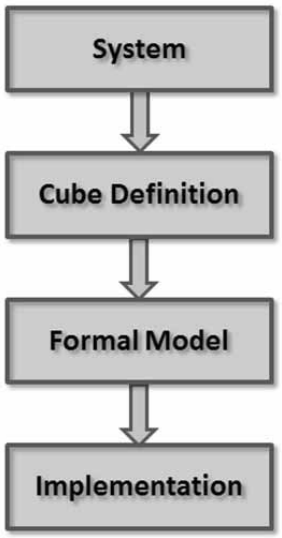


Figure 2. Analysis of the system (a variety of systems and their generalisation) leads us to the general "cube concept". This helps to formalise the real world and its control as well as future models. A formal model definition and implementation finalise the project phase.

Most important at this stage was the demand, that the cube concept should be as generic as possible not including specific model restrictions at that time. For these demands ontologies seem to fit in some kind of way. For this reason - and as a next step - the basic idea of ontologies, as well as the motivation for using such ontological analysis in modelling and the role in the modelling processes should be described.

3 Ontologies in Modelling

After all analysis of the requirements for cubes showed that ontological analysis could be a promising approach. The project team thought at that time, that probably the project will not need the whole range of possibilities, but some aspects seemed promising. Ontologies have been an effective tool in modelling and simulation to help to address some aspects in complex modelling & simulation projects.

To understand principles of the ontological approach and to estimate benefits and motivations for using Ontologies in modelling we relied on the work of Benjamin et al "Using Ontologies for Simulation Modeling" [8]. An ontology is an inventory of the kinds of entities that exist in a domain, their characteristic properties, and the relationships that can hold between them [9]. In our case the domain is the part of the actual world, which is a production plant. Such a production plant has its own ontology, which we refer as a domain ontology with some sub domains. In a domain ontology, we define various kinds of objects (e.g., machines and tools), properties (e.g., being made of metal), and relations between kinds and their instances (e.g., part of).

In general we need to extract the nature of concepts and relations in any domain and representing this knowledge in a structured manner. An ontology and its building differs from traditional modelling activities (adding information and data to a formal system description) not only in depth but also in breadth of the information used. As Benjamin et al describe in [8]: "Thus, an ontology development exercise will expand beyond asserting the mere existence of relations in a domain; the relations are "axiomatized" within an ontology (i.e., the behaviour of the relation is explicitly documented). Ontology development is motivated not so much by the search for knowledge for its own sake (as, ideally, in the natural and abstract sciences), but by the need to understand, design, engineer, and manage such systems effectively." For the cube concept, which should be used for various cube types within one model and as a basic library for future production plant models.

For defining ontologies different aspects are important as described in [10] especially determining the appropriate scope and granularity of ontologies and the use of ontologies as a basis for defining model repositories.

Inefficiency is often a problem in knowledge acquisition and management. Information that has been recorded before is captured again and modelling is done multiple times. Rather than having to identify information again and again in different applications, the idea of an ontology is to develop libraries " large revisable knowledge bases of structured, domain specific, ontological information in which can be put several uses for multiple application situations" [8].

The literature describes ontologies as important for modelling for a lot of reasons. Ontological analysis has been shown to be effective as a first step in the construction of robust knowledge based systems [11]. Modelling and simulation applications can take advantage of such technologies. As a second point, ontologies help to develop standard, reusable application and domain reference models. This characteristic seemed to fit for integration of various production plant types. Last but not least ontologies are at the heart of software systems that facilitate knowledge sharing.

Motivation for Using Ontologies in Modelling

Basic motivations for using ontologies in modelling and simulation are that they are useful across the modelling and simulation lifecycle, particularly in the problem analysis and conceptual model design phases. They play a critical role in simulation integration and simulation composability and they are important in facilitating simulation model interoperability, composition and information exchange.

One of the key ideas is to allow the decomposition of the overall system model into smaller, more manageable components, and to distribute the model development effort among different organisations or functional groups [12]. This is a perfect approach for the planned cube concept. Once the component simulation models have been developed, there is a need for mechanisms to assemble a simulation model of the entire target system in a manner that the "whole (system) = sum of its components."

An important challenge is modelling and simulation composability (from a set of independently developed components). "Composability is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements" [13]. Composability enables users to combine, recombine, and configure or reconfigure components in numerous ways to satisfy their diverse needs and requirements.

There are two forms of composability: syntactic and semantic . Syntactic composability deals with the compatibility of implementation details such as parameter passing mechanisms, external data accesses, and timing mechanisms. Semantic composability, on the other hand, deals with the validity and usefulness of composed simulation models [13].

As a matter of fact these advantages of ontological analysis seemed to perfectly fit the needs of our cube concept and the formal modelling process afterwards. The process described in Figure 2 was perfectly set for application of the basic ideas of ontological analysis.

Role of Ontologies in the Modelling Process

Simulation models are often designed to address a set of modelling objectives or to answer a set of questions. An important first step in simulation modelling is to define the purpose of the model. This activity involves several related activities. On one hand the developer gets a "list" of not formalised problem symptoms. The domain experts often describe a problem in terms of a list of observed symptoms or areas of concern. The desire is to identify the cause of these symptoms and to suggest remedies. As described in chapter 1 one of the main objectives for the cube approach is to introduce the possibility of bottom up foot printing for production plants and to identify the origin of those symptoms. In addition often the domain experts specify the objectives of a project in terms of a specific question that needs to be answered, or, alternatively, specifies explicit goals to be met. For instance, in our example the manager of the production plant might ask the question "How can I optimise my production process?" or state a goal: e.g., "I need to reduce used energy by 20% on all my machines.". Using clearly defined objectives can help a lot in both cases to formalise and structure the described goals.

The purpose of the model also depends on constraints on possible solutions to the problem. The domain expert, based on past experience with similar situations, often suggests a variety of possible alternative solutions that must be explored. For example, a production plant manager who would like to increase production rate may, because of a budgetary constraint, be unwilling to invest in new machines, but may instead be able to hire additional labour. Ontologies will help facilitate the above tasks as well.

The advantages and also the justification of investing additional resources needed for following an ontological approach instead of doing only the work which is unconditional are on one hand providing a mechanism to interpret and understand the problem descriptions. Domain experts often use specialised terminology to describe symptoms and problems. Domain ontologies help with the unambiguous interpretation of the problem statements and in precisely conveying information about the problem to the simulation modeller. Cube can – in a reduced way – fulfil these characteristics. In addition harmonizing statements of objects that are described from multiple perspectives (often, this is a non-trivial task because of terminological differences and the lack of explicit descriptions of the semantics of different terms and concepts – see also [8]). Last but not least the ontological analysis unambiguously interprets limiting constraints that need to be addressed relative to accomplishing project goals.

All together the BaMa Cube concept will not fulfil all formal needs and demands of an ontology. As a matter of fact within BaMa the ontological approach was identified to support various needs of the modelling process. It helps in the process of getting "axiomatized" rules for the modelling of production plant sub systems. So the behaviour of the relations between subsystems is explicitly documented as well as the possibility how and what to "footprint". Objects, properties and relations are clearly defined and are reproducible for every simulation project, that will be implemented with the cube concept. BaMa will not only generate "one model of one production plant" but will develop libraries and large revisable knowledge bases of structured, domain specific, ontological information in which can be put several uses for multiple application situations. In practice scope and granularity of the cubes can be defined clearly and can also be supervised. By using ontological analysis decomposition of the overall system model into smaller, more manageable components is done as well as distribution of the model development will be possible. The aim of composability enables future users of BaMa to combine, recombine, and configure or reconfigure components in numerous ways.

4 Cube Definition

On basis of the above described ideas the generic term "cube" describes an encapsulated part of the observed overall system (domain). This is part of a methodological approach to address the high system complexity and heterogeneity by dividing the overall system from an energetic point of view into well-defined manageable modules (see Figure 3), which then allow a focused sys-

tem analysis independent form the surrounding environment. Integrating different viewpoints and areas of engineering (machinery, energy system, building, and logistics) in a single system description can be interpreted as combining a number of ontological sub-domains and makes it necessary to establish a general specification of the cube properties and interfaces.

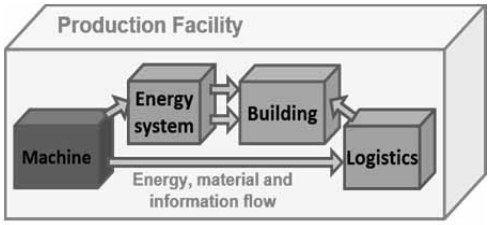


Figure 3. Production facility as interacting cubes.

The cubes consolidate all information and resource flows (energy, materials, etc.) within identical system boundaries, which not only promotes transparency during simultaneous analysis of energy and material flows, but the obtained modularity also increases flexibility for adaptation to specific environmental conditions.

Cubes have uniformly and consistently defined interfaces through which they interact with each other by exchanging energy, material and information flow, see Figure 2.

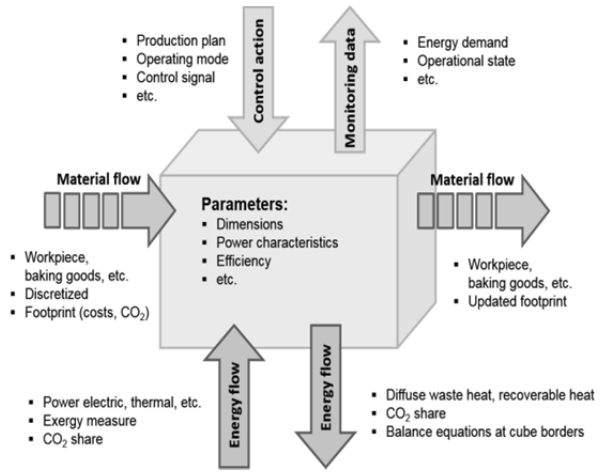


Figure 4. Generic cube interfaces with energy, material and information flows.

The material flow incorporates the immediate value stream (e.g. work piece, baking goods) and is described as discrete entities.

All necessary energy flow (electrical, thermal, etc.) is represented as continuous variables together with their respective CO2 rates and is quantified inside the cube boundaries using balance equations. Information flow provides operating states and monitoring values for the higher-level control as well as control actions for the cube module

This modular cube description and specified interfaces then enables analysing and modelling the internal behaviour independent from its surroundings. For experimental analysis based on measurement data, cube interfaces can be equipped with measuring devices to detect incoming and outgoing flows. Also, experimental production cubes are being constructed which allow a more in-depth energy analysis and the inclusion of more detailed measurement information for developing data models and usage in simulation.

The modularisation of the observed overall system is not only used for developing simulation models for these systems. So the cubes have not only the "virtual simulation block" (so-called virtual cube, see Figure 5) in the form of a component in a simulation model, which we have to formalise later on but also the representation in the "real world" e.g. in the automation system of the production plant.

The retained encapsulation and interaction via defined interfaces provides flexibility during internal modelling of the cubes (e.g. as mathematical models, data models, etc.) and for reusing implemented components in other models.

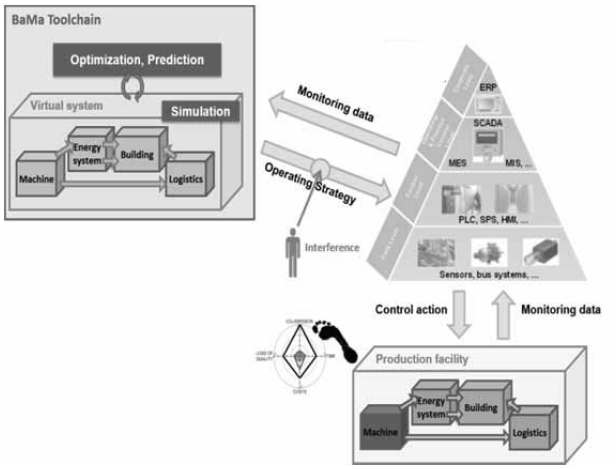


Figure 5. Architecture of the BaMa toolchain including the production facility in the "real world" and the a virtual representation of the observed system (simulation).

Figure 5 shows the relationship between real and virtual cubes in the simulation environment and the integration into the overall automation system architecture. The BaMa toolchain obtains measurement and status data from different levels of the automation system and on the other hand delivers prediction data and proposals for optimised operation strategies that can be adopted - with user interaction - in the real system.
The generic interface and attributes definition of the cubes serves as a basis for specifying four cube categories (see Figure 6).

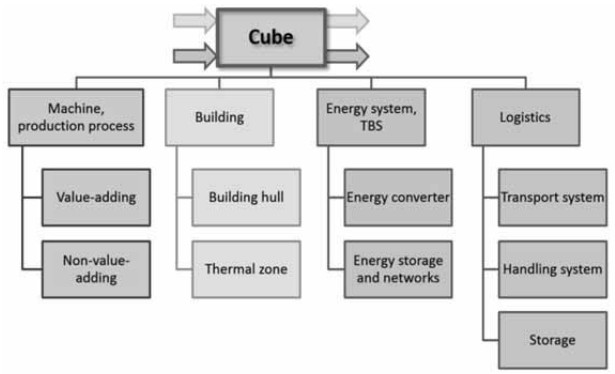


Figure 6. Categories and subcategories of cubes.

Defining the cubes succeeded in the possibility to have reusable modules for representing machines and all other physical inventory within a production plant. Both discrete and continuous flows can pass through the system. The modules are on basis of one methodology (all cubes are children from a master cube, see Figure 6) and can so be implemented in the same way. A more detailed description of the cube methodology and the individual cube categories can be found on the cube subsection on the BaMa project website with the address (http://bama.ift.tuwien.ac.at/). As a matter of fact while doing the cube concept, the modelling group of BaMa always had in mind how to formalise in the next step the model libraries on basis of the given features and interfaces, which was helpful in the next step.

5 Formalisation of Cubes

After the generic description of cubes the question of implementation arises. As far as described we combined various areas of production plants, where entities are able to pass from one area to the other. Still we need to be able to generate the planned foot printing.

As described in the last chapter on the one hand, the modelling approach needs to provide solutions for hybrid systems, i.e. systems containing continuous as well as discrete parts. Of course there are many software tools which offer solutions for either continuous or discrete models but not for combined models. Still, there are a few commonly known simulation environments like Simulink or Modelica who allow the combination of discrete and continuous model parts. In the case of Simulink, for example, discrete SimEvents models can be combined with continuous models described by ordinary differential equations (ODEs) where the SimEvents scheduler and the ODE solver work in parallel and cooperate, which seems to work fine for very simple trials, but as soon as large or rather complex systems are implemented, the simulation can fail due to non-resolvable errors. Additionally, the execution of actions intended to take place at the same time an event occurs has to be defined by the user right before or right after the event in order to prevent unintentional results.

On the other hand in the BAMA project buildings as well as machines, building services and logistics have to be modelled and simulated on the whole in spite of their different requirements regarding modelling approaches and simulation techniques. As this is virtually impossible to realise in one tool alone, the most common way to face this task is to use cooperative simulation (co-simulation). There exist some co-simulation tools developed especially for systems containing buildings and machines, but most of them regard mainly thermal processes and perhaps energy consumption but disregard resources and do not support optimisation. Furthermore these tools in general gravely restrict the software used for partial models.

These problems were approached by taking the step between the generic description (Cube Definition) and the actual Implementation - using a simulation formalism (Formal Model) – see Figure 2. In 1976 Bernard Zeigler proposed in his book “Theory of Modeling and Simulation” [14] a classification of dynamic system-models into three basic types: Discrete Event -, Discrete Time – and Differential Equation – systems (DEV, DTS, DES). DEV are usually simulated using an event-scheduler, DTS are system models where changes of state-values are happening in equidistant instances of time and DES as purely continuous models, described with differential equations. Zeigler introduced system-specification-formalism for all three types (DEVS, DTSS and DESS) where DTSS is a subtype of DEVS.

Very important properties of the formalisms are their hierarchical nature and their closure under coupling which perfectly fits the cube features. That is, an atomic model of each formalism has inputs and outputs, which can be coupled with inputs and outputs of other atomic blocks or with the inputs and outputs of an overlying non-atomic model which inhabits these atomic models (hierarchical). The resulting overlying model now behaves exactly like an atomic model (closure under coupling) of the particular formalism and therefore again can be coupled with other atomic and non-atomic models. In the following part we assume the knowledge of atomic and coupled DEVs and atomic and coupled DESs (see [14].

On basis of these atomic and coupled DEVS and DESS Zeigler introduced an additional formalism called DEV&DESS [15] standing for Discrete Event and Differential Equation System Specification. DEV&DESS is intended to describe so called hybrid system. In this context, hybrid system means a system consisting of both, a discrete and a continuous part, which is exactly what is needed for cubes. Atomic DEV&DESS systems can be described with the system $DEV\&DESS_{atomic} = < X^{discr}, X^{cont}, Y^{discr}, Y^{cont}, S, \delta_{ext}, C_{int}, \delta_{int}, \lambda^{discr}, f, \lambda^{cont} >$ where $X^{discr}, Y^{discr}, X^{cont}, Y^{cont}$ describes a set of possible discrete and continuous inputs and outputs and $S = S^{discr} \times S^{cont}$ is a set of possible states, which describes the state space. Together with $Q = \{(s^{discr}, s^{cont}, e) | s^{discr} \in S^{discr}, s^{cont} \in S^{cont}, e \in \mathbb{R}_0^+\}$ we get $\delta_{ext}: Q \times X^{cont} \times X^{discr} \to S$ and $\delta_{int}: Q \times X^{cont} \to S$ as internal and external state transition function, $\lambda^{discr}: Q \times X^{discr} \to Y^{discr}$ and $\lambda^{cont}: Q \times X^{cont} \to Y^{cont}$ as discrete and continuous output function as well as $f: Q \times X^{cont} \to S^{cont}$ as rate of change function ("right side" of an "ODE-System") and $C_{int}: Q \times X^{cont} \to \{true, false\}$ as state event condition function.

As described above the DESS and DEVS formalisms are well known in literature. In our case we focus on the additional meaning of C_{int}. C_{int} is a function of the actual state q and continuous input value $x^{cont}(t)$ and is responsible for triggering internal events, which then may cause a discrete output $y^{discr} = \lambda^{discr}(q, x^{cont})$ and definitely results in the execution of δ_{int}. Therefore, internal events in DEV&DESS are not exclusively dependable on time, as it is the case with DEVS, but may also be triggered because of the system state S reaching a certain threshold. Events of the later type are called state-events.
Since the state transition functions δ_{int} and δ_{ext} update the whole state, including its continuous part, they may lead to a discontinuous change in s^{cont}. Thus, as s^{cont} is the output of an integrator, this integrator needs to be reseted, each time an external or internal event occurs.

The last distinguishing feature of the whole, DEV&DESS, to its components DEVS and DESS is the dependency of δ_{int} and λ^{discr} of the actual continuous input value. For DEV&DESS to be well defined, we need to fulfil both, the requirements for the DEVS part, and the requirements for the DESS part. Therefore for each possible input-trajectories and initial states, during a finite time interval only a finite number of events is allowed to happen, the function f again has to meet the Lipschitz requirements and the continuous input and output signals need to be bounded and piecewise continuous.
Coupled DEV & DESS $N =< X^{discr} \times X^{cont}, Y^{discr} \times Y^{cont}, D, \{M_d\}_{d \in D}, \{I_d\}_{d \in D \cup \{N\}}, \{Z_d\}_{d \in D \cup \{N\}}, Select >$ are described via $X^{discr}, Y^{discr}, X^{cont}, Y^{cont}$ as a set of possible discrete and continuous inputs and outputs, D as a set of involved "child-DEV&DESS"-denominators, M_d as child DEV&DESS of N for each $d \in D$, $I_d \subset D \cup \{N\}$ influencer set of $d, d \notin I_d$and finally together with $Z_d$as interface map for d and $Select: 2^{D \cup \{N\}} \to D \cup \{N\}$as tie-breaking function we get the whole system.

The meaning of all the terms listed above are already known, either from the atomic DEVS definition or from coupled DEVS or coupled DESS systems. But there are some restrictions, concerning the coupling of discrete outputs with continuous inputs and vice versa. At first, we divide the interface map Z_d into two component functions. One for the calculation of the discrete inputs of block d $Z_d^{discr}: YX_i \to XY_d^{discr}$ and one for the calculation of the continuous inputs $Z_d^{cont}: YX_i \to XY_d^{cont}$. for each $i \in I_d$

Second, we need to define, how to interpret a connection from an discrete output to an continuous input and the other way round: Discrete output signals, actually are only existent at instance of time, where they are produced. The rest of the time, the value of the output-signal is the empty set $\emptyset$ or non existent. However, to enable connections between discrete outputs and continuous inputs, we define discrete outputs to be piecewise constant. So the value of a discrete output at a time between two output-events is always the value of the last output-event. Therefore it is allowed to connect dis-

crete outputs arbitrary to continuous inputs. The other way round isn't that easy, and it is necessary to apply restrictions. Thus, continuous outputs are only allowed to be connected to discrete inputs, if they are piecewise constant. One could think of a connection from discrete to continuous being realised by putting an additional DEV&DESS-block in between, that receives the discrete output at its discrete input and forwards it to its continuous output. The other way around works too.

As DEV&DESS sums up the functionality of both sides, the discrete and the continuous one, the modeller has to deal with the requirements of each formalism as well. On the one hand, the modeller needs to take care, not to produce algebraic loops and on the other hand he also needs to think of how to define the tie-breaking function select for the model to produce the desired behaviour. As Zeigler showed [15], all three basic formalism, DEVS, DTSS(already included in DEVS) and DESS describe subclasses of the set of DEV&DESS-describable systems. Therefore DEV&DESS-describable is perfectly suited to formally describe and simulator-independently hybrid models of real systems. In our case - as a step in between - we used the cube formalism as organisational structuring of the modelling process using ontological analysis know how. Every cube has continuous inputs like various forms of energy, which are part of a continuous model, and many cubes, like machine cubes handling work pieces, have discrete inputs which are handled in a discrete system part of the machine model.

Since the DEV&DESS formalism does not specify solution methods, solution algorithms for the discrete part and differential equation solvers for the continuous part can be chosen at the point of implementation. In the case of cubes comprising purely continuous models, the DESS formalism can be applied and still linked with other cubes described by DEV&DESS or DEVS for plain discrete systems. Additionally, several atomic DEV&DESS can be embraced by another DEV&DESS called coupled DEV&DESS afterward for even better structuring; hence the DEV&DESS formalism also fulfils the hierarchy requirement, which represents an obligatory demand in the BAMA cube definition.

As every DEV&DESS, be it coupled or atomic, can be regarded as separate systems and each DEV&DESS represents one cube in which the balance equations consider everything within the cube's borders, which are per definition balance borders, closure regarding balance equations can also be ensured as long as the generic description of the cube can guarantee it.

DEVS is a very general formalism. As a result, it can be shown, that a lot of other discrete-event-formalism, as for example Event-Graphs, State charts, Petri-Nets and even Cellular Automata describe subclasses of the set of all systems describable by DEVS. That's why Zeigler proposes the so called DEVS-Bus as common interface for multi-formalism simulation. For implementation and formalisation this keeps the possibility of a "general approach" for integrating domain experts knowledge in future approaches and involve possible additional model concepts (e.g. additional cubes shall be described in one of the ways mentioned above).

6 Implementation

Last but not least, since digital computers only are able to work in a discrete way, discretisation is necessary for each DEVS and DESS-part of a DEV&DESS to be able to be simulated on a digital computer. For pure DESS-models, usually ODE-solver-algorithms are used, to numerically solve the differential equations, i.e. to simulate the DESS model. Therefore, the DESS model in combination with the used ODE-solver constitutes a DEVS model, approximating the DESS model. This resulting DEVS model, as each DEVS model, can then be simulated error-free on a digital computer, apart from the error due to the finite representation of real numbers.

But due to the fact that the DEV&DESS formalism is, as its name implies, just a formalism, it is independent from the implementation software. This is very important for the BAMA project since a lot of participating industry partners already use certain automation software which is intended to be able to communicate with the simulation software and every developing partner has preferred simulation tools or limited licenses.

The DEV&DESS formalism does not restrict the possibilities for the cube interfaces. In the cube definition described briefly above it has been defined that input and output signals can be arrays and may represent physical values which carry a unit or other attributes ensuring consistency. This is possible with the DEV&DESS formalism since the only specification for inputs or outputs to a DEV&DESS is that there is a set of discrete and/or a set of continuous inputs and outputs. Hence the demands on cube interfaces can be met by the DEV&DESS formalism. Finally taking a deeper look at ontological analysis was worth doing, even if BaMa did not implement its own ontology. Defining and implementing the process as described below (see Figure 7) was one of the keys to successfully implement the cube methodology in the first phase of BaMa.

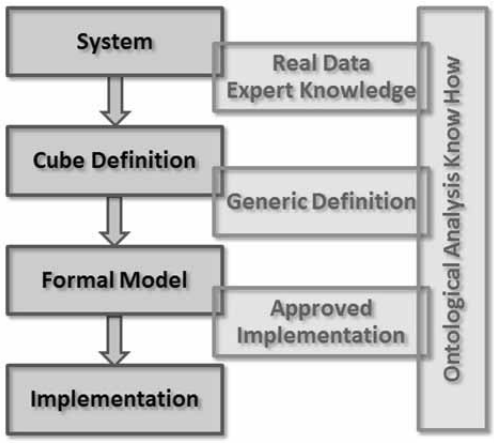


Figure 7. System Analysis and Modelling uses Ontological Analysis Knowhow for reusable, quality assured results.

At the actual point of BaMa the definition of the DEV&DESS formalism is finalised. As a matter of fact there is still a link missing to get to the implementation itself, but on the one hand there exist several tools implementing the DEV&DESS formalism with a certain approach like PowerDEVS using QSS for the discretisation of the DESS parts and thus transforming DEV&DESS into DEVS only, QSS-Solver with the Micro-Modelica language, M/CD++, or a Simulink library for DEV&DESS developed at the Hochschule Wismar or DEVS-only tools like DEVS-Suite, CD++ and JDEVS; on the other hand in the course of the BAMA project several typical scenarios have already been formalised with the DEV&DESS formalism and implemented PowerDEVS for test purposes, so it is warranted that this formalism can actually be used as a bridge from the BAMA cube definition to the BAMA implementation.

References

[1] S. Thiede, G. Bogdanski, C. Herrmann, "A systematic method for increasing the energy and resource efficiency in manufacturing companies" Procedia CIRP, Vol. 2, pp 28 – 33, 2012

[2] F. Bleicher, F. Dür, I. Leobner, I. Kovacic, B. Heinzl, W. Kastner, „Co-simulation environment for optimizing energy efficiency in production systems" CIRP Annals – Manufacturing Technology, Vol. 63, pp 441 – 444, 2014

[3] K. Bunse, M. Vodicka, P. Schönsleben, M. Brülhart, F. Ernst, „Integrating energy efficiency performance in production management - gap analysis between industrial needs and scientific literature" Journal of Cleaner Production, Vol. 19, pp. 667 – 679, 2011

[4] E. Bonneville, A. Rialhe, "Good practice for energy efficiency in industry", http://www.leonardo-energy.org/sites/leonardo-energy/files/root/Documents/2009/DSM-industry.pdf (last accessed Dezember 15, 2014).

[5] J. P. Padgett, A. C. Steinemann, J. H. Clarke, M. P. Vandenbergh, "A comparison of carbon calculators", Environmental Impact Assessment Review 28, 2008, pp 106–115.

[6] M. Gaussin, G.Hub, S.Abolghasem, S.Basu, M.R. Shankar, B.Bidanda, "Assessing the environmental footprint of manufactured products: A survey of current literature", Int.J.Production Economics 146, 2013,pp. 515–523.

[7] R. Neugebauer, E. Westkämper, F. Klocke, A. Kuhn, M. Schenk, A. Michaelis, D. Spath, E. Weidner, "Energieeffizienz in der Produktion - Untersuchung zu Handlungs- und Forschungsbedarf", FhG, 2008, pp. 350-351.

[8] P. Benjamin, M. Patki, R. Mayer, "Using Ontologies for Simulation Modeling". Proceedings of the 2006 Winter Simulation Conference, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds., pp. 1151-1159, 2006

[9] P. Benjamin, C. Menzel, and R. J. Mayer.Towards a method for acquiring CIM ontologies. International Journal of Computer Integrated Manufacturing 1995, 8 (3), 225–234.

[10] P. Fishwick and J. Miller. Ontologies for modeling and simulation: Issues and approaches. Proceedings of 2004 Winter Simulation Conference. Piscataway, New Jersey: Institute for Electrical and Electronics Engineers.

[11] J. Hobbs, W. Croft, T. Davies, D. Edwards, and K. Laws. The TACITUS Commonsense Knowledge Base, Artificial Intelligence Research Center, SRI International, 1987

[12] P. Benjamin and M. Graul. A framework for adaptive modeling and ontology-driven simulation. Proceedings of SPIE, Enabling Technologies for Simulation Science X, Vol.6227, 2006

[13] M.D. Petty and E. W. Weisel. A composability lexicon. Proceedings of the Spring 2003 Simulation Interoperability Workshop , Orlando FL, March 30- April 4 2003, pp. 181-187.

[14] B. Zeigler B. "Theory of Modeling and Simulation", Wiley-Interscience Publication, John Wiley & Sons, 1976

[15] B. Zeigler, H. Praehofer and T.G. Kim, "Theory of Modelling and Simulation", Second Edition:, Elsevier Academic Press, 2000.

Acknowledgement. The project BAMA (Balanced Manufacturing) is funded by the Austrian Research Promotion Agency (FFG, project number 840746) and the Austrian Klima- und Energiefonds (KLIEN).

SNE Simulation News

EUROSIM Data and Quick Info

EUROSIM 2016

9th EUROSIM Congress on Modelling and Simulation

City of Oulu, Finland, September 16-20, 2016

www.eurosim.info

Contents

Simulation Notes Europe SNE is the official membership journal of EUROSIM and distributed / available to members of the EUROSIM Societies as part of the membership benefits. **SNE** is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With **Online SNE** the publisher **ARGESIM** follows the **Open Access** strategy for basic **SNE** contributions. Since 2012 **Online SNE** contributions are identified by DOI 10.11128/sne.xx.nnnnn. for better web availability and indexing.

Print SNE, high-resolution **Online SNE**, and additional **SNE** contributions are available via membership in a **EUROSIM** society.

This *EUROSIM Data & Quick Info* compiles data from EUROSIM societies and groups: addresses, weblinks, officers of societies with function and email, to be published regularly in **SNE** issues.

SNE Editorial Office /ARGESIM

→ *www.sne-journal.org, www.eurosim.info*
office@sne-journal.org (info, news)
eic@sne-journal.orgt Felix Breitenecker (publications)

If you have any information, announcement, etc. you want to see published, please contact a member of the editorial board in your country or the editorial office. For scientific publications, please contact the EiC.

EUROSIM
Federation of European Simulation Societies

General Information. **EUROSIM**, the Federation of European Simulation Societies, was set up in 1989. The purpose of **EUROSIM** is to provide a European forum for simulation societies and groups to promote advancement of modelling and simulation in industry, research, and development. → *www.eurosim.info*

Member Societies. **EUROSIM** members may be national simulation societies and regional or international societies and groups dealing with modelling and simulation. At present **EUROSIM** has fourteen *Full Members* and three *Observer Members*:

ASIM	Arbeitsgemeinschaft Simulation *Austria, Germany, Switzerland*
CEA-SMSG	Spanish Modelling and Simulation Group *Spain*
CROSSIM	Croatian Society for Simulation Modeling *Croatia*
CSSS	Czech and Slovak Simulation Society *Czech Republic, Slovak Republic*
DBSS	Dutch Benelux Simulation Society *Belgium, Netherlands*
FRANCOSIM	Société Francophone de Simulation *Belgium, France*
HSS	Hungarian Simulation Society *Hungary*
ISCS	Italian Society for Computer Simulation *Italy*
LIOPHANT	LIOPHANT Simulation Club *Italy & International, Observer Member*
LSS	Latvian Simulation Society *Latvia*
PSCS	Polish Society for Computer Simulation *Poland*
SIMS	Simulation Society of Scandinavia *Denmark, Finland, Norway, Sweden*
SLOSIM	Slovenian Simulation Society *Slovenia*
UKSIM	United Kingdom Simulation Society *UK, Ireland*
KA-SIM	Romanian Society for Modelling and Simulation, *Romania, Observer Member*
ROMSIM	Romanian Society for Modelling and Simulation, *Romania, Observer Member*
RNSS	Russian National Simulation Society *Russian Federation, Observer Member*

EUROSIM Board / Officers. EUROSIM is governed by a board consisting of one representative of each member society, president and past president, and representatives for SNE Simulation notes Europe. The President is nominated by the society organising the next **EUROSIM** Congress. Secretary and Treasurer are elected out of members of the Board.

President	Esko Juuso (SIMS) *esko.juuso@oulu.fi*
Past President	Khalid Al.Begain (UKSIM) *kbegain@glam.ac.uk*
Secretary	Borut Zupančič (SLOSIM) *borut.zupancic@fe.uni-lj.si*
Treasurer	Felix Breitenecker (ASIM) *felix.breitenecker@tuwien.ac.at*
SNE Repres.	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*

SNE – Simulation Notes Europe. SNE is a scientific journal with reviewed contributions as well as a membership newsletter for **EUROSIM** with information from the societies in the *News Section*. **EUROSIM** societies are offered to distribute to their members the journal **SNE** as official membership journal. **SNE** Publishers are **EUROSIM**, ARGESIM and ASIM.

Editor-in-chief	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*

→ *www.sne-journal.org*,
✉ *office@sne-journal.org*

EUROSIM Congress. EUROSIM is running the triennial conference series **EUROSIM** Congress. The congress is organised by one of the **EUROSIM** societies.

EUROSIM 2016 will be organised by SIMS in Oulu, Finland, September 16-20, 2016.

Chairs / Team EUROSIM 2016

Esko Juuso EUROSIM President, *esko.juuso@oulu.fi*
Erik Dahlquist SIMS President, *erik.dahlquist@mdh.se*
Kauko Leiviskä EUROSIM 2016 Chair,
kauko.leiviska@oulu.fi

→ *www.eurosim.info*
✉ *office@automaatioseura.fi*

EUROSIM Member Societies

ASIM
German Simulation Society
Arbeitsgemeinschaft Simulation

ASIM (Arbeitsgemeinschaft Simulation) is the association for simulation in the German speaking area, servicing mainly Germany, Switzerland and Austria. ASIM was founded in 1981 and has now about 700 individual members, and 30 institutional or industrial members.

→ *www.asim-gi.org* with members' area

info@asim-gi.org, *admin@asim-gi.org*

ASIM – Inst. f. Analysis and Scientific Computing
Vienna University of Technology
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria

ASIM Officers	
President	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*
Vice presidents	Sigrid Wenzel, *s.wenzel@uni-kassel.de*
	T. Pawletta, *pawel@mb.hs-wismar.de*
Secretary	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
Treasurer	Anna Mathe, *anna.mathe@tuwien.ac.at*
Membership Affairs	S. Wenzel, *s.wenzel@uni-kassel.de*
	W. Maurer, *werner.maurer@zhwin.ch*
	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
Universities / Research Inst.	S. Wenzel, *s.wenzel@uni-kassel.de*
	W. Wiechert, *W.Wiechert@fz-juelich.de*
	J. Haase, *Joachim.Haase@eas.iis.fraunhofer.de*
	Katharina Nöh, *k.noeh@fz-juelich.de*
Industry	S. Wenzel, *s.wenzel@uni-kassel.de*
	K. Panreck, *Klaus.Panreck@hella.com*
Conferences	Klaus Panreck *Klaus.Panreck@hella.com*
	J. Wittmann, *wittmann@htw-berlin.de*
Publications	Th. Pawletta, *pawel@mb.hs-wismar.de*
	Christina Deatcu, *christina.deatcu@hs-wismar.de*
	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
Repr. EUROSIM	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
	N. Popper, *niki.popper@drahtwarenhandlung.at*
Education / Teaching	A. Körner, *andreas.koerner@tuwien.ac.at*
	N. Popper, *niki.popper@drahtwarenhandlung.at*
	Katharina Nöh, *k.noeh@fz-juelich.de*
International Affairs	A. Körner, *andreas.koerner@tuwien.ac.at*
	O. Rose, *Oliver.Rose@tu-dresden.de*
Editorial Board SNE	T. Pawletta, *pawel@mb.hs-wismar.de*
	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
Web EUROSIM	Anna Mathe, *anna.mathe@tuwien.ac.at*

Last data update December 2013

ASIM Working Committee. **ASIM**, part of GI - Gesellschaft für Informatik, is organised in Working Committees, dealing with applications and comprehensive subjects in modelling and simulation:

ASIM Working Committee	
GMMS	Methods in Modelling and Simulation Th. Pawletta, *pawel@mb.hs-wismar.de*
SUG	Simulation in Environmental Systems Wittmann, *wittmann@informatik.uni-hamburg.de*
STS	Simulation of Technical Systems H.T.Mammen, *Heinz-Theo.Mammen@hella.com*
SPL	Simulation in Production and Logistics Sigrid Wenzel, *s.wenzel@uni-kassel.de*
EDU	Simulation in Education/Education in Simulation N. Popper, *niki.popper@dwh.at* A. Körner, *andreas.koerner@tuwien.ac.at*
	Working Groups for Simulation in Business Administration, in Traffic Systems, for Standardisation, for Validation, etc.

CEA-SMSG – Spanish Modelling and Simulation Group

CEA is the Spanish Society on Automation and Control In order to improve the efficiency and to deep into the different fields of automation, the association is divided into thematic groups, one of them is named 'Modelling and Simulation', constituting the group.

→ *www.cea-ifac.es/wwwgrupos/simulacion*

→ *simulacion@cea-ifac.es*

CEA-SMSG / María Jesús de la Fuente,
System Engineering and AutomaticControl department,
University of Valladolid,
Real de Burgos s/n., 47011 Valladolid, SPAIN

CAE - SMSG Officers	
President	M. À. Piera Eroles, *MiquelAngel.Piera@uab.es*
Vice president	Emilio Jiminez, *emilio.jiminez@unirioja.es*
Repr. EUROSIM	Emilio Jiminez, *emilio.jiminez@unirioja.es*
Edit. Board SNE	Emilio Jiminez, *emilio.jiminez@unirioja.es*
Web EUROSIM	Mercedes Peres, *mercedes.perez@unirioja.es*

Last data update December2013

CROSSIM – Croatian Society for Simulation Modelling

CROSSIM-*Croatian Society for Simulation Modelling* was founded in 1992 as a non-profit society with the goal to promote knowledge and use of simulation methods and techniques and development of education. CROSSIM is a full member of **EUROSIM** since 1997.

→ *www.eurosim.info*

vdusak@foi.hr

✉ CROSSIM / Vesna Dušak
Faculty of Organization and
Informatics Varaždin, University of Zagreb
Pavlinska 2, HR-42000 Varaždin, Croatia

CROSSIM Officers	
President	Vesna Dušak, *vdusak@foi.hr*
Vice president	Jadranka Božikov, *jbozikov@snz.hr*
Secretary	Vesna Bosilj-Vukšić, *vbosilj@efzg.hr*
Executive board members	Vlatko Čerić, *vceric@efzg.hr*
	Tarzan Legović, *legovic@irb.hr*
Repr. EUROSIM	Jadranka Božikov, *jbozikov@snz.hr*
Edit. Board SNE	Vesna Dušak, *vdusak@foi.hr*
Web EUROSIM	Jadranka Bozikov, *jbozikov@snz.hr*

Last data update December2012

CSSS – Czech and Slovak Simulation Society

CSSS -The *Czech and Slovak Simulation Society* has about 150 members working in Czech and Slovak national scientific and technical societies (*Czech Society for Applied Cybernetics and Informatics*, *Slovak Society for Applied Cybernetics and Informatics*). The main objectives of the society are: development of education and training in the field of modelling and simulation, organising professional workshops and conferences, disseminating information about modelling and simulation activities in Europe. Since 1992, CSSS is full member of **EUROSIM**.

→ *www.fit.vutbr.cz/CSSS*

snorek@fel.cvut.cz

✉ CSSS / Miroslav Šnorek, CTU Prague
FEE, Dept. Computer Science and Engineering,
Karlovo nam. 13, 121 35 Praha 2, Czech Republic

CSSS Officers	
President	Miroslav Šnorek, *snorek@fel.cvut.cz*
Vice president	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Treasurer	Evžen Kindler, *ekindler@centrum.cz*
Scientific Secr.	A. Kavička, *Antonin.Kavicka@upce.cz*
Repr. EUROSIM	Miroslav Šnorek, *snorek@fel.cvut.cz*
Deputy	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Edit. Board SNE	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Web EUROSIM	Petr Peringer, peringer@fit.vutbr.cz

Last data update December2012

DBSS – Dutch Benelux Simulation Society

The Dutch Benelux Simulation Society (DBSS) was founded in July 1986 in order to create an organisation of simulation professionals within the Dutch language area. DBSS has actively promoted creation of similar organisations in other language areas. DBSS is a member of **EUROSIM** and works in close cooperation with its members and with affiliated societies.

→ www.eurosim.info

a.w.heemink@its.tudelft.nl

✉ DBSS / A. W. Heemink
Delft University of Technology, ITS - twi,
Mekelweg 4, 2628 CD Delft, The Netherlands

DBSS Officers	
President	A. Heemink, *a.w.heemink@its.tudelft.nl*
Vice president	W. Smit, *smitnet@wxs.nl*
Treasurer	W. Smit, *smitnet@wxs.nl*
Secretary	W. Smit, *smitnet@wxs.nl*
Repr. EUROSIM	A. Heemink, *a.w.heemink@its.tudelft.nl*
Deputy	W. Smit, *smitnet@wxs.nl*
Edit. Board SNE	A. Heemink, *a.w.heemink@its.tudelft.nl*

Last data update April 2006

FRANCOSIM – Société Francophone de Simulation

FRANCOSIM was founded in 1991 and aims to the promotion of simulation and research, in industry and academic fields. Francosim operates two poles.

- Pole Modelling and simulation of discrete event systems. Pole Contact: *Henri Pierreval*, *pierreva@imfa.fr*
- Pole Modelling and simulation of continuous systems. Pole Contact: *Yskandar Hamam*, *y.hamam@esiee.fr*

→ *www.eurosim.info*

y.hamam@esiee.fr

✉ FRANCOSIM / Yskandar Hamam
Groupe ESIEE, Cité Descartes,
BP 99, 2 Bd. Blaise Pascal,
93162 Noisy le Grand CEDEX, France

FRANCOSIM Officers	
President	Karim Djouani, *djouani@u-pec.fr*
Treasurer	François Rocaries, *f.rocaries@esiee.fr*
Repr. EUROSIM	Karim Djouani, *djouani@u-pec.fr*
Edit. Board SNE	Karim Djouani, *djouani@u-pec.fr*

Last data update December2012

HSS – Hungarian Simulation Society

The Hungarian Member Society of EUROSIM was established in 1981 as an association promoting the exchange of information within the community of people involved in research, development, application and education of simulation in Hungary and also contributing to the enhancement of exchanging information between the Hungarian simulation community and the simulation communities abroad. HSS deals with the organization of lectures, exhibitions, demonstrations, and conferences.

→ *www.eurosim.info*

javor@eik.bme.hu

✉ HSS / András Jávor,
Budapest Univ. of Technology and Economics,
Sztoczek u. 4, 1111 Budapest, Hungary

HSS Officers	
President	András Jávor, *javor@eik.bme.hu*
Vice president	Gábor Szűcs, *szucs@itm.bme.hu*
Secretary	Ágnes Vigh, *vigh@itm.bme.hu*
Repr. EUROSIM	András Jávor, *javor@eik.bme.hu*
Deputy	Gábor Szűcs, *szucs@itm.bme.hu*
Edit. Board SNE	András Jávor, *javor@eik.bme.hu*
Web EUROSIM	Gábor Szűcs, *szucs@itm.bme.hu*

Last data update March 2008

ISCS – Italian Society for Computer Simulation

The Italian Society for Computer Simulation (ISCS) is a scientific non-profit association of members from industry, university, education and several public and research institutions with common interest in all fields of computer simulation.

→ *www.eurosim.info*

Mario.savastano@uniina.at

✉ ISCS / Mario Savastano,
c/o CNR - IRSIP,
Via Claudio 21, 80125 Napoli, Italy

ISCS Officers	
President	M. Savastano, *mario.savastano@unina.it*
Vice president	F. Maceri, *Franco.Maceri@uniroma2.it*
Repr. EUROSIM	F. Maceri, *Franco.Maceri@uniroma2.it*
Secretary	Paola Provenzano, *paola.provenzano@uniroma2.it*
Edit. Board SNE	M. Savastano, *mario.savastano@unina.it*

Last data update December2010

LIOPHANT Simulation

Liophant Simulation is a non-profit association born in order to be a trait-d'union among simulation developers and users; Liophant is devoted to promote and diffuse the simulation techniques and methodologies; the Association promotes exchange of students, sabbatical years, organization of International Conferences, organization of courses and stages in companies to apply the simulation to real problems.

→ *www.liophant.org*

info@liophant.org

✉ LIOPHANT Simulation, c/o Agostino G. Bruzzone,
DIME, University of Genoa, Polo Savonese,
via Molinero 1, 17100 Savona (SV), Italy

LIOPHANT Officers	
President	A.G. Bruzzone, *agostino@itim.unige.it*
Director	E. Bocca, *enrico.bocca@liophant.org*
Secretary	A. Devoti, *devoti.a@iveco.com*
Treasurer	Marina Masseimassei@itim.unige.it
Repr. EUROSIM	A.G. Bruzzone, *agostino@itim.unige.it*
Deputy	F. Longo, *f.longo@unical.it*
Edit. Board SNE	F. Longo, *f.longo@unical.it*
Web EUROSIM	F. Longo, *f.longo@unical.it*

Last data update December2013

LSS – Latvian Simulation Society

The Latvian Simulation Society (LSS) has been founded in 1990 as the first professional simulation organisation in the field of Modelling and simulation in the post-Soviet area. Its members represent the main simulation centres in Latvia, including both academic and industrial sectors.

→ *briedis.itl.rtu.lv/imb/*

merkur@itl.rtu.lv

LSS / Yuri Merkuryev, Dept. of Modelling and Simulation Riga Technical University Kalku street 1, Riga, LV-1658, LATVIA

LSS Officers	
President	Yuri Merkuryev, *merkur@itl.rtu.lv*
Secretary	Artis Teilans, *Artis.Teilans@exigenservices.com*
Repr. EUROSIM	Yuri Merkuryev, *merkur@itl.rtu.lv*
Deputy	Artis Teilans, *Artis.Teilans@exigenservices.com*
Edit. Board SNE	Yuri Merkuryev, *merkur@itl.rtu.lv*
Web EUROSIM	Oksana Sosho, *oksana@itl.rtu.lv*

Last data update December2013

PSCS – Polish Society for Computer Simulation

PSCS was founded in 1993 in Warsaw. PSCS is a scientific, non-profit association of members from universities, research institutes and industry in Poland with common interests in variety of methods of computer simulations and its applications. At present PSCS counts 257 members.

→ *www.ptsk.man.bialystok.pl*

leon@ibib.waw.pl

PSCS / Leon Bobrowski, c/o IBIB PAN, ul. Trojdena 4 (p.416), 02-109 Warszawa, Poland

PSCS Officers	
President	Leon Bobrowski, *leon@ibib.waw.pl*
Vice president	Tadeusz Nowicki, *Tadeusz.Nowicki@wat.edu.pl*
Treasurer	Z. Sosnowski, *zenon@ii.pb.bialystok.pl*
Secretary	Zdzislaw Galkowski, *Zdzislaw.Galkowski@simr.pw.edu.pl*
Repr. EUROSIM	Leon Bobrowski, *leon@ibib.waw.pl*
Deputy	Tadeusz Nowicki, *tadeusz.nowicki@wat.edu.pl*
Edit. Board SNE	Zenon Sosnowski, z.sosnowski@pb.ed.pl
Web EUROSIM	Magdalena Topczewska *m.topczewska@pb.edu.pl*

Last data update December2013

SIMS – Scandinavian Simulation Society

SIMS is the *Scandinavian Simulation Society* with members from the four Nordic countries Denmark, Finland, Norway and Sweden. The SIMS history goes back to 1959. SIMS practical matters are taken care of by the SIMS board consisting of two representatives from each Nordic country (Iceland one board member).

SIMS Structure. SIMS is organised as federation of regional societDjouaniies. There are FinSim (Finnish Simulation Forum), DKSIM (Dansk Simuleringsforening) and NFA (Norsk Forening for Automatisering).

→ *www.scansims.org*

esko.juuso@oulu.fi

SIMS / Esko Juuso, Department of Process and Environmental Engineering, 90014 Univ.Oulu, Finland

SIMS Officers	
President	Esko Juuso, *esko.juuso@oulu.fi*
Vice president	Erik Dahlquist, *erik.dahlquist@mdh.se*
Treasurer	Vadim Engelson, *vadim.engelson@mathcore.com*
Repr. EUROSIM	Esko Juuso, *esko.juuso@oulu.fi*
Edit. Board SNE	Esko Juuso, *esko.juuso@oulu.fi*
Web EUROSIM	Vadim Engelson, *vadim.engelson@mathcore.com*

Last data update December2013

SLOSIM – Slovenian Society for Simulation and Modelling

SLOSIM - Slovenian Society for Simulation and Modelling was established in 1994 and became the full member of **EUROSIM** in 1996. Currently it has 69 members from both slovenian universities, institutes, and industry. It promotes modelling and simulation approaches to problem solving in industrial as well as in academic environments by establishing communication and cooperation among corresponding teams.

→ www.slosim.si

slosim@fe.uni-lj.si

SLOSIM / Rihard Karba, Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

SLOSIM Officers	
President	Vito Logar, *vito.logar@fe.uni-lj.si*
Vice president	Božidar Šarler, *bozidar.sarler@ung.si*
Secretary	Aleš Belič, *ales.belic@sandoz.com*
Treasurer	Milan Simčič, *milan.simcic@fe.uni-lj.si*
Repr. EUROSIM	B. Zupančič, *borut.zupancic@fe.uni-lj.si*
Deputy	Vito Logar, *vito.logar@fe.uni-lj.si*
Edit. Board SNE	Rihard Karba, *rihard.karba@fe.uni-lj.si*
Web EUROSIM	Vito Logar, *vito.logar@fe.uni-lj.si*

Last data update December2013

UKSIM - United Kingdom Simulation Society

UKSIM has more than 100 members throughout the UK from universities and industry. It is active in all areas of simulation and it holds a biennial conference as well as regular meetings and workshops.

→ *www.uksim.org.uk*

david.al-dabass@ntu.ac.uk

✉ UKSIM / Prof. David Al-Dabass
Computing & Informatics,
Nottingham Trent University
Clifton lane, Nottingham, NG11 8NS
United Kingdom

UKSIM Officers	
President	David Al-Dabass, *david.al-dabass@ntu.ac.uk*
Vice president	A. Orsoni, *A.Orsoni@kingston.ac.uk*
Secretary	Richard Cant, *richard.cant@ntu.ac.uk*
Treasurer	A. Orsoni, *A.Orsoni@kingston.ac.uk*
Membership chair	K. Al-Begain, *kbegain@glam.ac.uk*
Univ. liaison chair	R. Cheng, *rchc@maths.soton.ac.uk*
Repr. EUROSIM	Richard Zobel, *r.zobel@ntlworld.com*
Deputy	K. Al-Begain, *kbegain@glam.ac.uk*
Edit. Board SNE	Richard Zobel, *r.zobel@ntlworld.com*

Last data update December2013

EUROSIM OBSERVER MEMBERS

KA-SIM Kosovo Simulation Society

Kosova Association for Modeling and Simulation (KA – SIM, founded in 2009), is part of Kosova Association of Control, Automation and Systems Engineering (KA – CASE). KA – CASE was registered in 2006 as non Profit Organization and since 2009 is National Member of IFAC – International Federation of Automatic Control. KA-SIM joined EUROSIM as Observer Member in 2011.

KA-SIM has about 50 members, and is organizing the international conference series International Conference in Business, Technology and Innovation, in November, in Durrhes, Albania, an IFAC Simulation workshops in Pristina.

→ *www.ubt-uni.net/ka-case*

ehajrizi@ubt-uni.net

✉ MOD&SIM KA-CASE
Att. Dr. Edmond Hajrizi
Univ. for Business and Technology (UBT)
Lagjja Kalabria p.n., 10000 Prishtina, Kosovo

KA-SIM Officers	
President	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Vice president	Muzafer Shala, *info@ka-sim.com*
Secretary	Lulzim Beqiri, *info@ka-sim.com*
Treasurer	Selman Berisha, *info@ka-sim.com*
Repr. EUROSIM	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Deputy	Muzafer Shala, *info@ka-sim.com*
Edit. Board SNE	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Web EUROSIM	Betim Gashi, *info@ka-sim.com*

Last data update December2013

ROMSIM – Romanian Modelling and Simulation Society

ROMSIM has been founded in 1990 as a non-profit society, devoted to theoretical and applied aspects of modelling and simulation of systems. ROMSIM currently has about 100 members from Romania and Moldavia.

→ *www.ici.ro/romsim/*

sflorin@ici.ro

✉ ROMSIM / Florin Stanciulescu,
National Institute for Research in Informatics, Averescu Av. 8 – 10, 71316 Bucharest, Romania

ROMSIM Officers	
President	Florin Stanciulescu, *sflorin@ici.ro*
Vice president	Florin Hartescu, *flory@ici.ro* Marius Radulescu, *mradulescu@ici.ro*
Repr. EUROSIM	Florin Stanciulescu, *sflorin@ici.ro*
Deputy	Marius Radulescu, *mradulescu@ici.ro*
Edit. Board SNE	Florin Stanciulescu, *sflorin@ici.ro*
Web EUROSIM	Zoe Radulescu, *radulescu@ici.ro*

Last data update December2012

RNSS – Russian Simulation Society

NSS - The Russian National Simulation Society (Национальное Общество Имитационного Моделирования – НОИМ) was officially registered in Russian Federation on February 11, 2011. In February 2012 NSS has been accepted as an observer member of **EUROSIM**.

→ *www.simulation.su*

yusupov@iias.spb.su

RNSS / R. M. Yusupov,
St. Petersburg Institute of Informatics and Automation RAS, 199178, St. Petersburg, 14th lin. V.O, 39

RNSS Officers	
President	R. M. Yusupov, *yusupov@iias.spb.su*
Chair Man. Board	A. Plotnikov, *plotnikov@sstc.spb.ru*
Secretary	M. Dolmatov, *dolmatov@simulation.su*
Repr. EUROSIM	R. M. Yusupov, *yusupov@iias.spb.su*
Deputy	B. Sokolov, *sokol@iias.spb.su*
Edit. Board SNE	Y. Senichenkov, *sneyb@dcn.infos.ru*

Last data update February 2012

SNE – Simulation Notes Europe

Simulation Notes Europe publishes peer reviewed *Technical Notes*, *Short Notes* and *Overview Notes* on developments and trends in modelling and simulation in various areas and in application and theory. Furthermore **SNE** documents the **ARGESIM Benchmarks** on *Modelling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions (*Benchmark Notes*). Special *Educational Notes* present the use of modelling and simulation in and for education and for e-learning.

SNE is the official membership journal of **EUROSIM**, the Federation of European Simulation Societies. A News Section in **SNE** provides information for **EUROSIM** Simulation Societies and Simulation Groups. In 2013, SNE introduced an extended submission strategy i) individual submissions of scientific papers, and ii) submissions of selected contributions from conferences of EUROSIM societies for post-conference publication (suggested by conference organizer and authors) – both with peer review.

SNE is published in a printed version (Print ISSN 2305-9974) and in an online version (Online ISSN 2306-0271). With **Online SNE** the publisher **ARGESIM** follows the **Open Access** strategy, allowing download of published contributions for free. Since 2012 **Online SNE** contributions are identified by an DOI (Digital Object Identifier) assigned to the publisher **ARGESIM** (DOI prefix 10.11128). **Print SNE**, high-resolution **Online SNE**, source codes of the *Benchmarks* and other additional sources are available for subscription via membership in a **EUROSIM** society.

Authors Information. Authors are invited to submit contributions which have not been published and have not being considered for publication elsewhere to the SNE Editorial Office. SNE distinguishes different types of contributions (*Notes*):

- *Overview Note* – State-of-the-Art report in a specific area, up to 14 pages, only upon invitation
- *Technical Note* – scientific publication on specific topic in modelling and simulation, 6 – 8 (10) pages
- *Education Note* – modelling and simulation in / for education and e-learning; max. 6 pages
- *Short Note* – recent development on specific topic, max. 4 pages
- *Software Note* – specific implementation with scientific analysis, max 4 pages
- *Benchmark Note* – Solution to an ARGESIM Benchmark;basic solution 2 pages, extended and commented solution 4 pages, comparative solutions on invitation

Interested authors may find further information at **SNE's** website → **www.sne-journal.org** (layout templates for *Notes*, requirements for benchmark solutions, etc.).

SNE Editorial Office / ARGESIM

→ *www.sne-journal.org, www.eurosim.info*

office@sne-journal.org (info, news)

eic@sne-journal.org Felix Breitenecker (publications)

EUROSIM 2016

9th EUROSIM Congress on Modelling and Simulation

City of Oulu, Finland, September 12 – 16, 2016

EUROSIM Congresses are the most important modelling and simulation events in Europe. For EUROSIM 2016, we are soliciting original submissions describing novel research and developments in the following (and related) areas of interest: Continuous, discrete (event) and hybrid modelling, simulation, identification and optimization approaches. Two basic contribution motivations are expected: M&S Methods and Technologies and M&S Applications. Contributions from both technical and non-technical areas are welcome.

Congress Topics The EUROSIM 2016 Congress will include invited talks, parallel, special and poster sessions, exhibition and versatile technical and social tours. The Congress topics of interest include, but are not limited to:

- Intelligent Systems and Applications
- Hybrid and Soft Computing
- Data & Semantic Mining
- Neural Networks, Fuzzy Systems & Evolutionary Computation
- Image, Speech & Signal Processing
- Systems Intelligence and Intelligence Systems
- Autonomous Systems
- Energy and Power Systems
- Mining and Metal Industry
- Forest Industry
- Buildings and Construction
- Communication Systems
- Circuits, Sensors and Devices
- Security Modelling and Simulation
- Bioinformatics, Medicine, Pharmacy and Bioengineering
- Water and Wastewater Treatment, Sludge Management and Biogas Production
- Condition monitoring, Mechatronics and maintenance
- Automotive applications
- e-Science and e-Systems
- Industry, Business, Management, Human Factors and Social Issues
- Virtual Reality, Visualization, Computer Art and Games
- Internet Modelling, Semantic Web and Ontologies
- Computational Finance & Economics
- Simulation Methodologies and Tools
- Parallel and Distributed Architectures and Systems
- Operations Research
- Discrete Event Systems
- Manufacturing and Workflows
- Adaptive Dynamic Programming and Reinforcement Learning
- Mobile/Ad hoc wireless networks, mobicast, sensor placement, target tracking
- Control of Intelligent Systems
- Robotics, Cybernetics, Control Engineering, & Manufacturing
- Transport, Logistics, Harbour, Shipping and Marine Simulation

Congress Venue / Social Events The Congress will be held in the City of Oulu, Capital of Northern Scandinavia. The main venue and the exhibition site is the Oulu City Theatre in the city centre. Pre and Post Congress Tours include Arctic Circle, Santa Claus visits and hiking on the unique routes in Oulanka National Park.

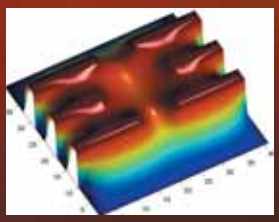
Congress Team: The Congress is organised by SIMS - Scandinavian Simulation Society, FinSim - Finnish Simulation Forum, Finnish Society of Automation, and University of Oulu.

Esko Juuso EUROSIM President, Erik Dahlquist SIMS President, Kauko Leiviskä EUROSIM 2016 Chair

Info: www.eurosim.info, office@automaatioseura.fi

Parlez-vous MATLAB?

Über eine Million Menschen weltweit sprechen MATLAB. Ingenieure und Wissenschaftler in allen Bereichen – von der Luft- und Raumfahrt über die Halbleiterindustrie bis zur Biotechnologie, Finanzdienstleistungen und Geo- und Meereswissenschaften – nutzen MATLAB, um ihre Ideen auszudrücken. Sprechen Sie MATLAB?



Modellierung eines elektrischen Potentials in einem Quantum Dot.

Dieses Beispiel finden Sie unter: www.mathworks.de/ltc

Image: Kim Young-Sang, Jung HeeJun, Quantum Device Lab, Hanyang Univ.

MATLAB®

The language of technical computing