TOWARDS A STANDARD COMPUTATIONAL REPRESENTATION FOR SYSTEM ENTITY STRUCTURES

Bikash Chandra Karmokar

Leibniz University Hannover
Welfengarten 1
30167 Hannover, Germany
bikash.chandra.karmokar@stud.uni-hannover.de

Umut Durak

German Aerospace Center (DLR)
Lilienthalplatz 7
38108 Braunschweig, Germany
umut.durak@dlr.de

Sven Hartmann

Clausthal University of Technology
Julius-Albert-Strasse 4
38678 Clausthal-Zellerfeld, Germany
sven.hartmann@tu-clausthal.de

Bernard P. Ziegler

RTSync Corp. and
Arizona Center for Integrative M&S
University of Arizona
1230 E. Speedway Blvd.
Tucson, AZ 85721-0104
zeigler@ece.arizona.edu

ABSTRACT

System Entity Structure (SES) is a high-level ontology which was introduced for knowledge representation of decomposition, taxonomy and coupling of systems. It has its roots from the systems theory-based approaches to modeling and simulation. SES has been applied for various purposes by modeling and simulation community, however, there still exists a lack of standardized computational representation. This hinders the shareability of SES artifacts and interoperability of SES tools. In search for wider acceptance and eventual standardization, this paper proposes a computational representation and supporting application agnostic tool suite: SESEditor and PESEditor.

Keywords: System Entity Structure (SES), XML Schema, XPath.

1 INTRODUCTION

The System Entity Structure (SES) is a high level ontology framework targeted to modeling, simulation, systems design and data engineering (Zeigler and Hammonds 2007). An SES is a formal structure governed by a set of elements and a small number of axioms that provide clarity and consistency to its models. It supports hierarchical and modular compositions allowing large complex structures to be built in step wise fashion from smaller, simpler ones. The axioms and functionality based semantics of the SES promote practical design and are easily understandable by data modelers (Salas 2008). SES is used in many applications for structural knowledge representation. These include creating suites of models for global warming (Zeigler et al. 2013), modeling the elements of a scenario in a research flight simulator (Durak et al. 2017), reengineering simulation models (Durak 2015) or metamodeling (Durak et al. 2017) or variant modeling in model-based design (Pawletta, Pascheka, and Schmidt 2015). It is particularly suitable for describing

system configurations (Deatcu et al. 2018). The theory is extended many times based on particular requirements of the applications (Santucci et al. 2016, Pawletta et al. 2016, Schmidt et al. 2016, Durak et al. 2018, Deatcu et al. 2018). Various SES tools have been developed, some of which are MATLAB/Simulink SES Toolbox (Pawletta, Pascheka, and Schmidt 2015), SESBuilder (Cheon, Kim, and Zeigler 2008), and SESToPy (Deatcu, Pawletta, and Folkerts 2019).

Despite its diverse use cases, there still exists a lack of standardized computational representation for the SES. This hinders the shareability of SES artifacts and interoperability of SES tools. In search for wider acceptance and eventual standardization, this paper proposes a computational representation and supporting application agnostic tool suite: SESEditor and PESEditor.

2 BACKGROUND

SES is a high-level ontology which was introduced for knowledge representation of decomposition, taxonomy and coupling of systems (Kim, Lee, Christensen, and Zeigler 1990). It has a set of elements and axioms. The elements of SES are Entity, Aspect, Specialization and Multiple-Aspect (Zeigler and Hammonds 2007). Real or artificial system components can be represented using Entity nodes. An Entity is an object of interest, and can also have variables attached to it. An Aspect denotes the decomposition relationship of an Entity node. Specialization nodes represent the taxonomy of an entity. A Multi-Aspect is a special kind of aspect, which represents a multiplicity relationship that specifies the parent entity as a composition of multiple entities of the same type. Aspect, Specialization and Multi-Aspect are represented by one, two and three vertical lines respectively. There are six axioms of SES: (1) uniformity, (2) strict hierarchy, (3) alternating mode, (4) valid brothers, (5) attached variables, and (6) inheritance (Zeigler 1984). According to the uniformity axiom, any two nodes with the same labels have isomorphic subtrees. Strict hierarchy defines a restriction that prevents a label from appearing more than once down any path of the tree. Alternating mode requires that, if a node is an Entity, then the successor is either Aspect or Specialization and vice versa. Valid brothers axiom disallows two brothers from having the same label. An attached variable specifies a constraint that variable types attached to the same item shall have distinct names. With inheritance, it is indicated that Specialization inherits all variables and Aspects.

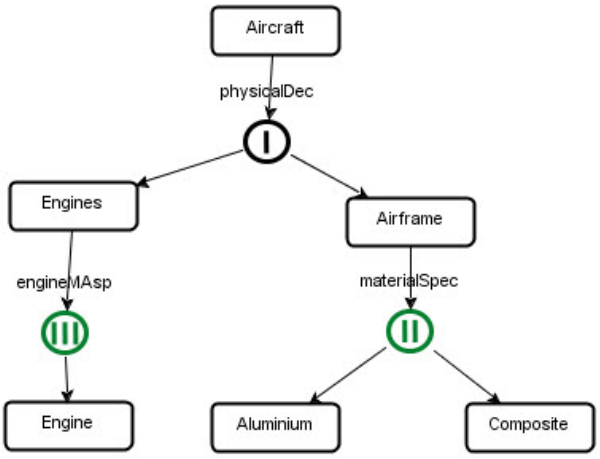

Figure 1: Aircraft metamodel using SES.

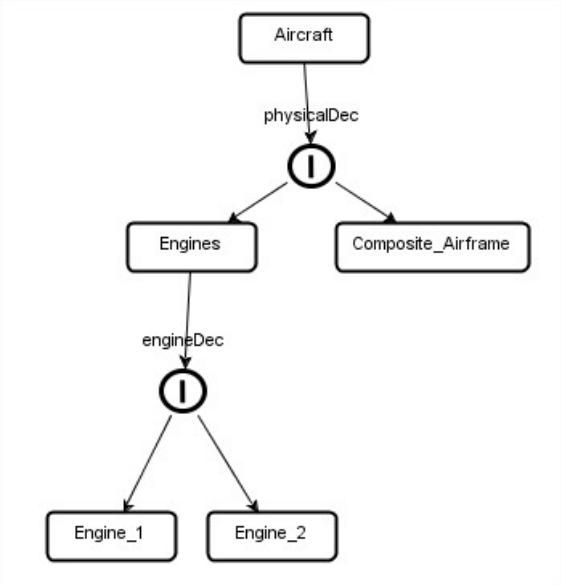

Figure 2: Pruned model of aircraft.

Pruning is the operation in which a unique system structure is derived from an SES and the result is called Pruned Entity Structure (PES). SES represents a family of models for a given application domain in terms of decompositions, component taxonomies and coupling specifications. In the modeling process, using SES, all the available options of a system are considered. As an SES describes a number of system configurations, the SES tree needs to be pruned to get a particular configuration. Pruning cuts off unnecessary structure from an SES tree based on the specification of a realistic frame to bring this particular configuration or PES which is a selection-free tree. The pruning process normally reduces an SES by removing choices for an entity which has multiple aspects and specializations consisting of multiple entities. An SES tree can be pruned by assigning values to the variables, choosing a particular subject from several aspect nodes for several decompositions of the system on the same hierarchical level, selecting one entity from various options of specialization node and specifying cardinality in Multi-Aspect node. Figure 1 shows an Aircraft metamodel using SES and Figure 2 shows the pruned model of the Aircraft metamodel.

3 COMPUTATIONAL REPRESENTATION

Computational representation means denoting SES and PES in a machine readable format. This paper is based on the XML based computational representation proposed by Zeigler and Hammonds (Zeigler and Hammonds 2007). As depicted in Figure 3 there are two important operations in the conceptual space. First, a particular SES is built using the constructs, structure and axioms of SES, called an SES Ontology. Then, Pruned Entity Structures (PESs) are obtained from this particular SES via pruning operations. In the computational space Zeigler and Hammonds propose that we can represent the SES ontology with an XML Schema and the particular SES as an XML file. They then recommend writing XML that specifies a particular SES to an XML Schema. Finally, this schema is then used during pruning for constructing and validating PESs. PESs are eventually XML instances.

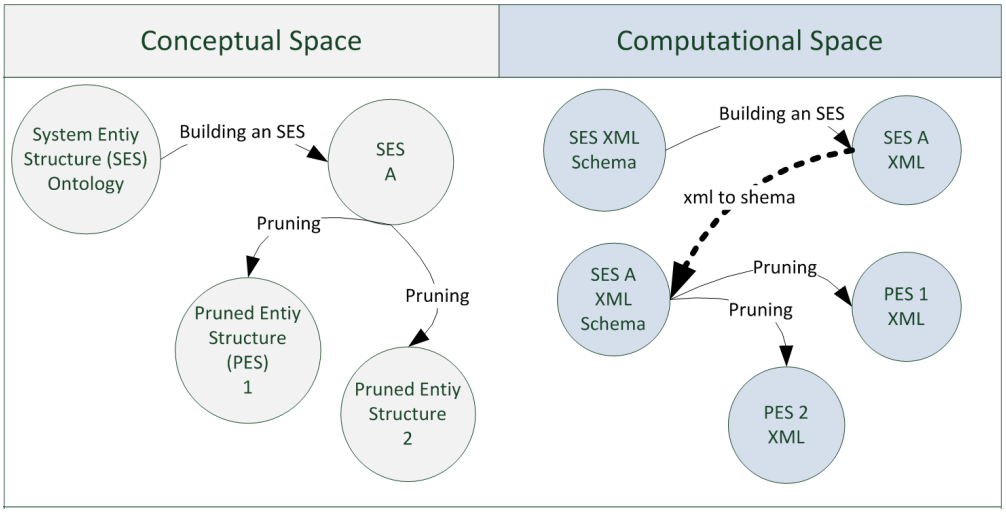

Figure 3: Computational representation.

The schema for the SES ontology can be specified using the XML Schema Definition Language (XSD). XSD provides means for describing the structure and the constraints of an XML document (Thompson et al. 2009). Referring back to Figure 3, the XML Schema for the SES ontology provides us a metasyntax at the metamodeling level. The Data Type Definition (DTD) representation of the SES ontology that is proposed by Zeigler and Hammonds (Zeigler and Hammonds 2007) is adapted to prepare the XML Schema representation that is presented in Figure 4. It reflects the elements of an SES and their structure as it is presented in the previous section.

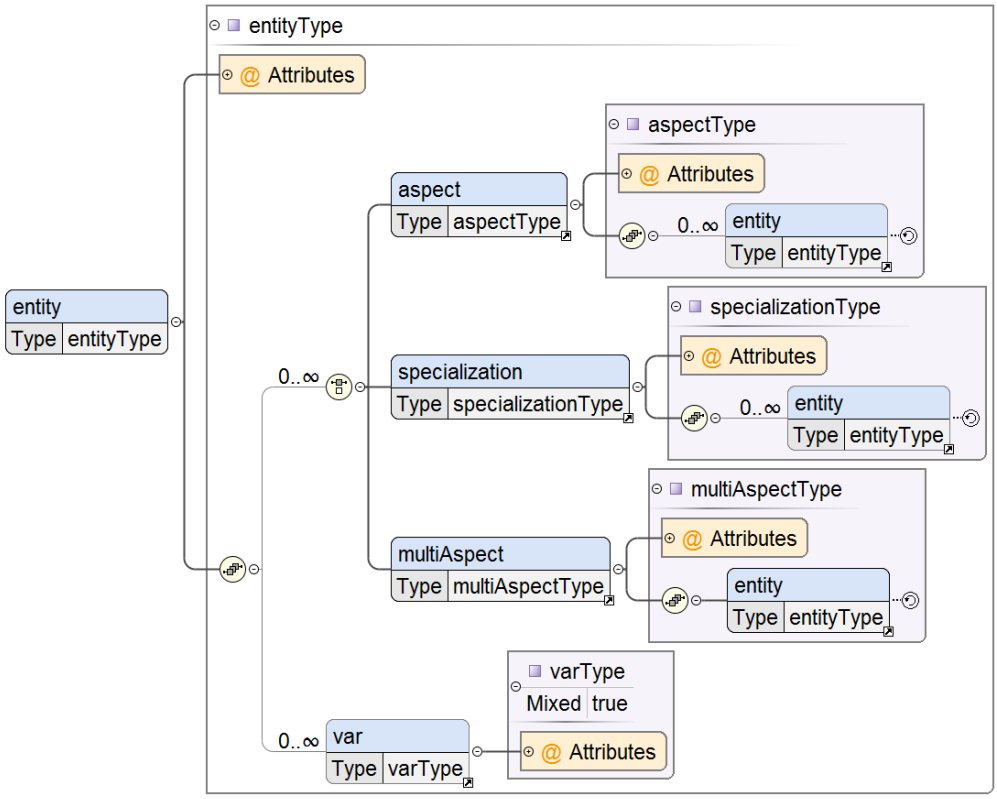

Figure 4: XML Schema for the SES ontology.

Extending what is proposed by Zeigler and Hammonds (Zeigler and Hammonds 2007), we use XML Schema capabilities for formal checking of SES against its axioms. XML Schema assert (xs:assert) is utilized to formalize the axioms of the SES ontology that are introduced at the previous section. Assertions provide means to constrain the existence and values of elements and attributes in XML Schemas. Assertions are specified using XPath 2.0 which is a functional language that is used to navigate through elements and attributes (Berglund et al. 2010).

A sample axiom for valid brothers is given below in Listing 1:

```
<xs:unique name="validBrothers">
  <xs:selector xpath="*/entity"/>
  <xs:field xpath="@name"/>
</xs:unique>
```

Listing 1: XPath for the valid brothers axiom.

A particular SES is then an XML document that conforms to the XML Schema for the SES ontology given in Figure 4. Following the Figure 3, the XML representation can be written into an XML Schema either programmatically or manually. SES Entity can be modeled as XML Schema element (xs:element) with a name attribute that designates the SES Entity name. It is by definition an XML Schema complex type (xs:complexType) which can contain elements and/or attributes. Aspect can also be represented by an XML Schema element (xs:element) with a name attribute that designates the SES Aspect name. It will also be a complex type which has an XML Schema sequence (xs:sequence) composed of XML Schema elements (xs:element). MultiAspect can be represented pretty much the same way as the Aspect, but with only having

SES Element	XSD Structure

Entity	`<xs:element name="[entity.name]">`
	`  <xs:complexType>`
	`  . . .`
	`  </xs:complexType>`
	`</xs:element>`

Aspect	`<xs:element name="[apect.name]">`
	`  <xs:complexType>`
	`  <xs:sequence>`
	`  <xs:element name="[child.entity.name]"/>`
	`  . . .`
	`  </xs:sequence>`
	`  </xs:complexType>`
	`</xs:element>`

multiAspect	`<xs:element name="[multiAspect.name]">`
	`  <xs:complexType>`
	`  <xs:sequence>`
	`  <xs:element name="[entity.name]"`
	`  minOccurs="[numberComponentsVar.min]"`
	`  maxOccurs="[numberComponentsVar.max]">`
	`  . . .`
	`  </xs:element>`
	`  </xs:sequence>`
	`  </xs:complexType>`
	`</xs:element>`

Specialization	`<xs:element name="[spec.name]">`
	`  <xs:complexType>`
	`  <xs:choice>`
	`  <xs:element name="[child.entity.name]"/>`
	`  . . .`
	`  </xs:choice>`
	`  </xs:complexType>`
	`</xs:element>`

Variable	`<xs:element name="[variable.name]">`
	`  <xs:complexType mixed="true">`
	`  <xs:attribute name="[var.name]"type="xs:+[var.rangeSpec]"/>`
	
	`  </xs:complexType>`
	`</xs:element>`

Figure 5: XML Schema for an SES.

one element in its sequence. The MultiAspect element possesses minOccurs and maxOccurs properties to specify the cardinality. Specialization is accordingly an element with a complex type. XML Schema choice (xs:choice) is then used to specify the specialization relation to child entities. Variable can be defined by attributes where attribute name is used to specify the name of a variable and type for its type. Figure 5 summarizes the XML Schema for an SES.

There is an overhead associated with representing an SES following the structure presented in Figure 5. A general restructuring is recommended by Zeigler and Hammonds (Zeigler and Hammonds 2007) for eliminating the elements including aspect, multiAspect and specialization. The semantics of aspect can be represented by an xs:sequence and mutiAspect with xs:sequence and minOccurs and maxOccurs attributes. xs:choice can then represent the semantic for the specialization. The reduced XML schema for an SES would be then as presented in Figure 6.

XML Schema assert (xs:assert) is further proposed to specify selection constraints in an SES. Selection constraints enable us to restrict the choices during pruning. Pretty much the same way as the axioms are defined at the XML Schema for the SES ontology, assertions can be specified using XPath 2.0. A typical example could be given using the musical performance SES from Zeigler and Hammonds (Zeigler and Hammonds 2007) that is given in Figure 7. As not all combinations are likely, constraints can be added in the MusicPerformanceDec aspect node. Listing 2 shows the the constraint for Symphonic, Folk and Jazz

SES Element	XSD Extract

Entity	`<xs:element name="[entity.name]">`
	`<xs:complexType>`
	`. . .`
	`</xs:complexType>`
	`</xs:element>`

Aspect	`<xs:element name="[entity.name]" >`
	`<xs:complexType>`
	`<xs:sequence>`
	`<xs:element name="[child.entity.name]"/>`
	`. . .`
	`</xs:sequence>`
	`</xs:complexType>`
	`</xs:element>`

multiAspect	`<xs:element name="[element.name]">`
	`<xs:complexType>`
	`<xs:sequence>`
	`<xs:element name="[child.entity.name]"`
	`minOccurs="[numberComponentsVar.min]"`
	`maxOccurs="[numberComponentsVar.max]">`
	`</xs:sequence>`
	`</xs:complexType>`
	`</xs:element>`

Specialization	`<xs:element name="[entity.name]">`
	`<xs:complexType>`
	`<xs:choice>`
	`<xs:element name="[child.entity.name]"/>`
	`. . .`
	`</xs:choice>`
	`</xs:complexType>`
	`</xs:element>`

Variable	`<xs:complexType>`
	`<xs:attribute name="[var.name]"type="xs:+[var.rangeSpec]"/>`
	`</xs:complexType>`

Figure 6: Restructured XML Schema for an SES.

Musical Performance. If Symphonic is selected from the Music node then the only option from performer is Orchestra can be selected. Orchestra selected if the Music selection is Folk. In case of Jazz music performance Soloist can not be selected. Orchestra and SmallGroup are possible performer for Jazz music.

```
<xs:assert test="if(Music/*[@name='Symphonic']) then (Performer/*[@name=
'Orchestra']) else true()"
/>

<xs:assert test="if(Music/*[@name='Folk']) then (Performer/*[@name=
'SmallGroup']) or (Performer/*[@name='Soloist']) else true()"
/>

<xs:assert test="if(Music/*[@name='Jazz']) then (Performer/*[@name=
'Orchestra']) or (Performer/*[@name='SmallGroup']) else true()"
/>
```

Listing 2: Constraint example for symphonic, folk and jazz musical performances.

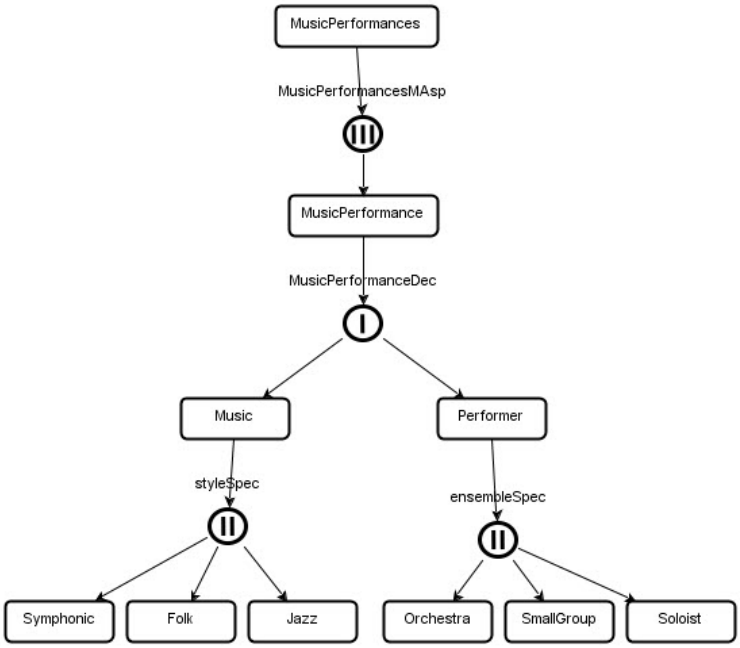

Figure 7: SES metamodel for musical performances.

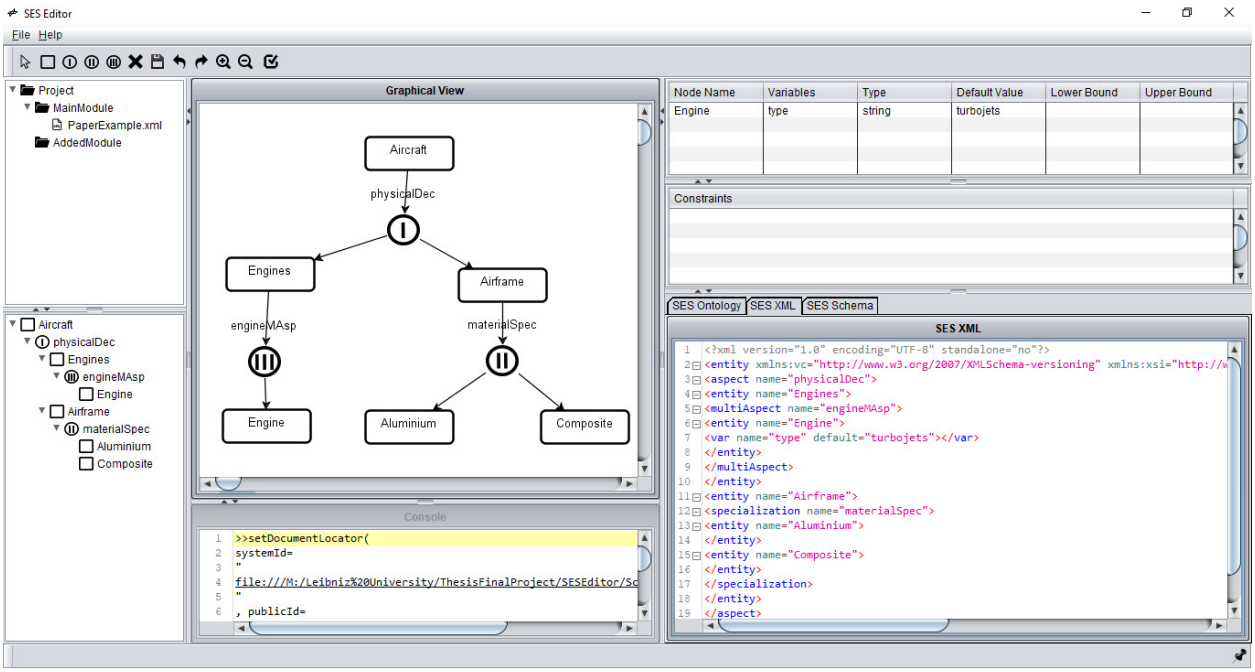

Figure 8: A screenshot from SESEditor.

4 APPLICATION AGNOSTIC SES TOOLS

4.1 SESEditor: The SES Modeling Environment

SESEditor has been developed as an application agnostic SES modeling environment. Figure 8 shows its Graphical User Interface (GUI) which is designed in such a way that a user can draw SES graphs on the screen almost as one would on paper. An SES is represented by a directed tree structure. Here, objects are represented by nodes that are connected using edges. In Figure 8, an SES model is presented in the editor where the root entity Aircraft is decomposed into Engines and Airframe using physicalDec aspect node. The Multi-Aspect node enginesMAsp decomposes Engines to Engine representing that Engines is made of multiple instances of Engine. Specialization node materialSpec is used to express that Airframe can be Aluminum or Composite.

Elements icons are added in the toolbox for easy access. The vertices or elements and edges can be drawn by clicking and dragging the mouse. Also nodes and edges can be easily moved to any position. The drawing panel is synchronized with the bottom-left tree. If there is an element addition in one place, either in the white drawing panel or the left tree, it will be added in both the sections automatically. Variables can be attached to the nodes. Eventually, the attached variables are listed in the variable table on the right top corner during any node selection from either of the trees. Furthermore, selection constraints can be added to the aspect node to restrict the choices of entities. These constraints are specified using XPath. Like variables, constraints are also listed on the constraint table on the right side of the editor during aspect node selection. The SESEditor allows the user to save part of the designed model as a module for future use. That saved module can be reused in any project later. SESEditor also has the ability to validate the created model against SES axioms using a predefined XML Schema. Validation results are displayed in the console window and for valid models, the SES XML and SES Schema are displayed in the bottom-right display window. The editor's export and import options increase the shareability of the designed models.

4.2 PESEditor: The Interactive PES Pruning Tool

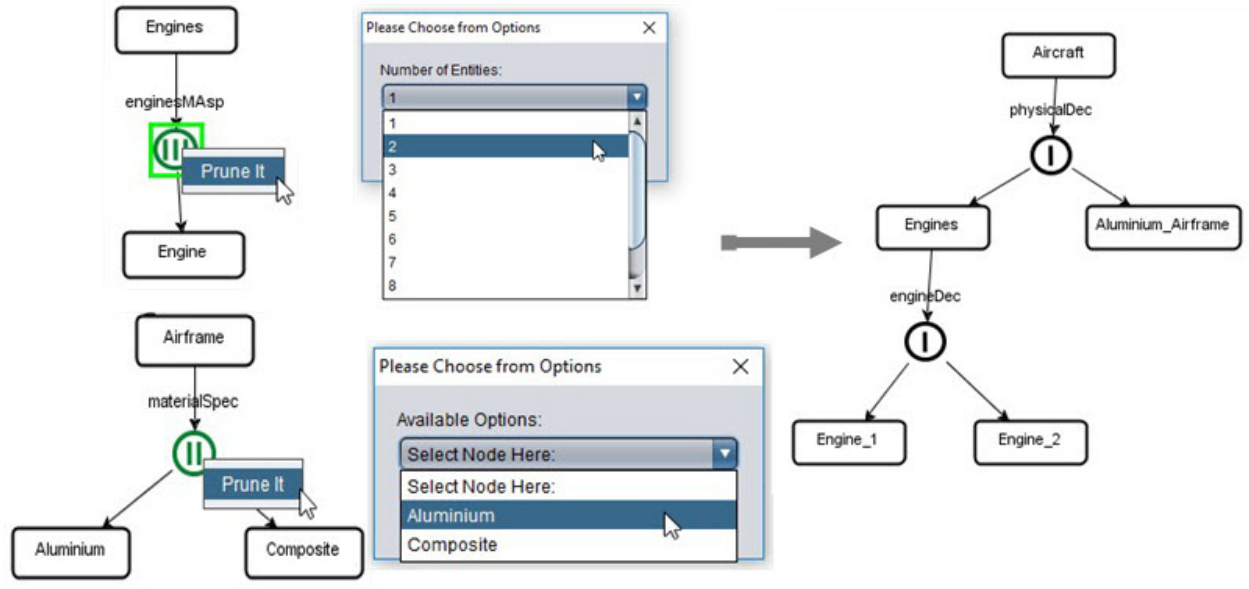

Figure 9: An excerpt from PESEditor.

PESEditor has been developed as an interactive pruning tool. The user basically selects each and every decision point, such as a specialization node, and resolves the decision. It supports all the design patterns proposed be Deatcu and colleagues (Deatcu et al. 2018) for interactive pruning of SES. Its GUI looks very similar to the SESEditor. It also has variable table for displaying variables or editing values of variables. Constraint table and console window work exactly the same way. Here, the left side tree is also synchronized with the white drawing panel and nodes are movable. Unlike SESEditor, here we can not create new projects, but only open SES models created in SESEditor. New elements cannot be added or deleted; the existing SES cannot be edited. The main functionality of PESEditor is interactive pruning of an SES model.

Figure 9 depicts an instance from interactive pruning using PESEditor. During pruning, MultiAspect node enginesMAsp is pruned and aspect node enginesDec is added with children Engine_1 and Engine_2 of the same type as Engine. From the available options, the specialization node is being pruned and Aluminum_-Airframe is being selected. Thus the completely pruned structure is created, where there are no choices left and a specific model of an Aircraft is shown. When the pruning is complete, PESEditor also supports transformations using XSLT. The user can transform the PES to the schema of choice without leaving the PESEditor.

5 CONCLUSION

Despite the wide utilization of System Entity Structures in modeling and simulation community, there is a lack of common computational representation. This eventually impairs shareability and interoperability. In search for wider acceptance and eventual standardization, based on previous work, this paper proposes a computational representation and introduces an application agnostic SES modeling environment, namely SESEditor and interactive pruning tool, namely PESEditor. Already existing XML based computational representation is revisited, and extended using XML Schema assert (xs:asset) to check the SES axioms and to specify selection constraints for a specifics SES. SESEditor and PESEditor is developed to demonstrate the utilization of this computational representation. SESEditor exemplified a user friendly way of constraint specification and PESEditor demonstrated an interactive pruning approach.

Though the proposed approach and the tooling is adding lots of benefits, the next challenge to tackle is the scalability. Both the graphical and the computational representations need to be enhanced to enable modular and hierarchical construction to handle large SES models. It includes but not limited to support for composition of SESs as in (Zeigler et al. 2013).

REFERENCES

Berglund, A., S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Simeon. 2010. *XML Path Language (XPath) 2.0*. World Wide Web Consortium.

Cheon, S., D. Kim, and B. P. Zeigler. 2008. "System Entity Structure for XML meta data modeling; application to the US climate normals". In *SEDE*, pp. 216–221.

Deatcu, C., H. Folkerts, T. Pawletta, and U. Durak. 2018. "Design patterns for variability modeling using SES ontology". In *Proceedings of the Model-driven Approaches for Simulation Engineering Symposium*. SCS.

Deatcu, C., T. Pawletta, and H. Folkerts. 2019. "MATLAB/Simulink's Variant Manager vs SESToPy". In *ASIM STS/GMMS Symposium - ARGESIM Report 57*, pp. 77–80.

Durak, U. 2015. "Extending the Knowledge Discovery Metamodel for architecture-driven simulation modernization". *Simulation* vol. 91 (12), pp. 1052–1067.

Durak, U., S. Jafer, R. Wittman, S. Mittal, S. Hartmann, and B. P. Zeigler. 2018. "Computational representation for a simulation scenario definition language". In *2018 AIAA Modeling and Simulation Technologies Conference*. AIAA.

Durak, U., T. Pawletta, H. Oguztuzun, and B. P. Zeigler. 2017. "System Entity Structure and Model Base framework in model based engineering of simulations for technical systems". In *Proceedings of the Symposium on Model-driven Approaches for Simulation Engineering*. SCS.

Durak, U., I. Pruter, T. Gerlach, S. Jafer, T. Pawletta, and S. Hartmann. 2017. "Using System Entity Structures to model the elements of a scenario in a research flight simulator". In *AIAA Modeling and Simulation Technologies Conference*. AIAA.

Kim, T.-G., C. Lee, E. R. Christensen, and B. P. Zeigler. 1990. "System Entity Structuring and Model Base management". *IEEE Transactions on Systems Man and Cybernetics* vol. 20 (5), pp. 1013–1024.

Pawletta, T., D. Pascheka, and A. Schmidt. 2015. "System Entity Structure ontology toolbox for MATLAB/Simulink: Used for Variant Modelling". In *Proc. of MATHMOD 2015-8th Vienna Int. Conf. on Mathematical Modelling*.

Pawletta, T., A. Schmidt, B. P. Zeigler, and U. Durak. 2016. "Extended variability modeling using System Entity Structure ontology within MATLAB/Simulink". In *Proceedings of the 49th Annual Simulation Symposium*. SCS.

Salas, M. C. 2008. "AutoDEVS: A methodology for automating systems development". *Electrical and Computer Engineering Dept., University of Arizona*.

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

Schmidt, A., U. Durak, and T. Pawletta. 2016. "Model-based testing methodology using System Entity Structures for Matlab/Simulink models". *Simulation* vol. 92 (8), pp. 729–746.

Thompson, H. S., N. Mendelsohn, D. Beech, and M. Maloney. 2009. "W3C XML schema definition language (XSD) 1.1 part 1: Structures". *The World Wide Web Consortium (W3C), W3C Working Draft Dec* vol. 3.

Zeigler, B. 1984. *Multifacetted modelling and discrete event simulation*. Academic Press.

Zeigler, B. P., and P. E. Hammonds. 2007. *Modeling and simulation-based data engineering: introducing pragmatics into ontologies for net-centric information exchange*. Elsevier.

Zeigler, B. P., C. Seo, R. Coop, and D. Kim. 2013. "Creating suites of models with System Entity Structure: Global warming example". In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium*. SCS.

AUTHOR BIOGRAPHIES

BIKASH CHANDRA KARMOKAR is a master student from Leibniz University Hannover. His research interest include simulation based data engineering, simulation in machine learning and artificial intelligence, software engineering, developing research tool and predictive modeling using machine learning. His email address is bikash.chandra.karmokar@stud.uni-hannover.de.

UMUT DURAK is a Research Scientist at Institute of Flight Systems of German Aerospace Center (DLR) and an adjunct faculty at the Department of Informatics of the Clausthal University of Technology. His research interests include model-driven approaches applied to engineering of simulation systems and modeling and simulation based engineering of flight systems. His email address is umut.durak@dlr.de.

SVEN HARTMANN is a Professor at the Department of Informatics of the Clausthal University of Technology. His research interests lie in information systems and databases in engineering, life sciences, and business. His email address is sven.hartmann@tu-clausthal.de.

BERNARD P. ZIEGLER is an Emeritus Professor at the University of Arizona and Adjunct Research Professor in the C4I Center at George Mason University. He is known for inventing Discrete Event System Specification (DEVS). He is currently participating in RTSync Corp., a developer of the MS4 modeling and simulation software based on DEVS, as the Chief Scientist. His email address is zeigler@ece.arizona.edu.