Systems Architecture Meta-Model for the MBSE Grid Framework

line 1: 1st Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*—Although Model-Based Systems Engineering (MBSE) might be understood and applied across diverse communities and industries differently, its fundamental principles and practices described in INCOSE Systems Engineering Handbook, are common. It is known that successful application of MBSE is impossible without a tool, language, and method as well as harmony among these three. The de-facto standard modeling language for MBSE is Systems Modeling Language (SysML). However, SysML is only a language, and it is not aligned with terminology systems engineers use daily in their work. This paper studies SysML as the standard language to model systems, and MBSE Grid as the framework to bridge the gap between Systems Engineering (SE) terminology described in the INCOSE Systems Engineering Handbook and SysML specification. It proposes a new systems engineering meta-model to describe a common SE terminology and bind it to the SysML concepts.**

Keywords—MBSE, SysML, MBSE Grid, meta-model, MBSE enablers

I. Introduction

SysML defines a few major concepts, like Block and Activity to capture structure and functionality of the system [15]. Neither of these terms exist in the Systems Engineering vocabulary used daily by systems engineers. That is one of the core challenges of applying SysML in industry without extending it with the needed terms. Extensions and customizations, however, are expensive to maintain [13].

MBSE Grid is one of but a few frameworks that use pure SysML to demonstrate how to model a system from requirements to solution [8]. It uses no extensions to the language. Nevertheless, it uses terminology aligned with *Systems Engineering Handbook V4* [5] and demonstrates how various terms in different SE lifecycle stages map to SysML stereotypes and diagrams. This mapping today is described in [1]. As the framework is used extensively in the industry today, we found a formal glossary and mapping between MBSE Grid terms and SysML concepts expressed as stereotypes missing. To formalize it, we propose a simplified systems engineering meta-model and its mapping to the SysML stereotypes.

The goal of this paper is to propose a common vocabulary for systems engineers, aligned with *Systems Engineering Handbook V4* [5] and MBSE Grid as well as mapped to SysML 1.6 [15]. The purpose of this common vocabulary is to bridge the gap between the language of systems engineers and SysML.

This paper is structured as follows: in Section 2, the related works are analyzed; in Section 3, the proposed meta-model is presented in detail; in Section 4, mappings from a sample model to the proposed meta-model, are provided; in Section 5, the achieved results, conclusions, and future work directions are indicated.

II. Related Works

Although MBSE is adopted widely in different industries, there are quite few generic approaches defining use of SysML to solve systems engineering challenges. In this section we analyze a few of them:

A. MBSE Grid

It is a pure SysML-compliant approach and requires no extensions to the standard SysML [8], [10]. MBSE Grid defines a few layers of abstraction that cover technical processes defined in ISO/IEC/IEEE 15288 standard [6], and unambiguously determines boundaries between them. There are three major layers also called domains: Problem, Solution, and Implementation [9]. Clear separation of layers of abstraction enables to allocate responsibilities for each task of MBSE.

The MBSE Grid framework is organized into a matrix layout [8], where rows rep-resent the above-described domains, and columns represent the four pillars of SysML introduced in [2]. The intersection of a row and a column represents a specific view of MBSE, which can be visualized in the form of a relevant SysML diagram.

B. Systems Architecture Framework

Inspired by the various enterprise architecture frameworks the System Architecture Framework (SAF) by the GfSE is a common, domain independent system architecture framework dedicated to support the MBSE of technical systems [3]. The SAF is a complementary framework to enterprise architecture frameworks, like Unified Architecture Framework (UAF) [14], supporting the needs of potential system suppliers to enterprise acquires.

SAF provides concept model and UML profile. Unlike MBSE Grid framework, SAF framework is based on extensions to Unified Modeling Language (UML) [16] and SysML. The

Identify applicable funding agency here. If none, delete this text box.

concept model provided by SAF is very extensive, capturing multiple different viewpoints and aspects of the System of Interest (SoI) development.

C. Unified Architecture Framework and other defense frameworks

The primary goal of UAF is to produce robust Enterprise Architecture Descriptions in alignment with ISO/IEC/IEEE 42010:2022 [7]. UAF provides a specification language that is readily understandable not only by the community of enterprise architects, architects of information technology systems or systems engineers, but also by a wide range of end users including executives and enterprise management that sponsor such systems, program managers that oversee their development, developers of supporting hardware and software (design, imple-mentation, and testing), subject matter experts, and end users [14].

A UAF Domain meta-model (DMM) is organized according to viewpoints. Thus, it is easy to understand which elements (including types, individuals, and tuples) can be used to build a specific view. The categorization of elements into types, individuals, and tuples is taken from IDEAS. In general, a UAF meta-model is a simplified version of complex 4D IDEAS ontology [4]. Although it is simplified, it is still powerful compared to the majority of existing enterprise modeling languages and methodologies. UAF, however, as NAF and DODAF are enterprise architecture frameworks. It means they are concerned of the wider picture than simply the architecture of a system. They address strategy, personnel, training of the personnel, logistics, how system integrates with other systems to achieve higher level goals, etc. EA frameworks require more extensive training and produce much larger models. It is not always a desired outcome. Moreover, UAF use UAFML, which is an extension of SysML just like SAF.

D. SysML Cookbook

The SE^2 team has provided their guidelines for modeling in *SE^2's Cookbook* [12]. SE^2 provided SysML model, illustrates the results of their comprehensive modeling. It provides collection of complex data structures that can be edited, augmented, queried, and reported on by means of a suitable tool. SE^2 model use set of SysML extensions in the form of stereotypes, however, they do not provide conceptual model to link SysML and extended concepts to it.

The SAF concept model and UAF DMM are both close to what we are looking for. Nevertheless, both are too extensive, and we found both missing a clear mapping to *Systems Engineering Hand-book V4*. Concepts in SAF are reused from various enterprise architecture frameworks, but not precisely *Systems Engineering Handbook V4*, e.g., problem and solution concepts are not addressed. What we try to achieve is way simpler meta-model mapped to both *Systems Engineering Handbook V4* and SysML. For this purpose, MBSE Grid is the framework providing the closest language to *Systems Engineering Handbook V4*.

III. Suggested Meta-Model

This section focuses on the detailed description of the systems architecture meta-model for the MBSE Grid framework. Prior to this, the MBSE Grid framework is briefly introduced, and means used to organize and define the meta-model are disclosed.

A. Introduction to MBSE Grid

The introduction of the MBSE Grid framework reveals the purpose of using the framework and provides a brief overview of its layout. The subsequently described modeling workflow also reveals how the framework aligns with technical processes described in the ISO/IEC/IEEE 15288 standard [6].

1) MBSE Grid Framework

It determines how to specify the system architecture starting from the blank sheet of paper, that is, from the problem domain analysis and definition, all the way until the optimal configuration of the system architecture is achieved. The MBSE Grid framework can be depicted as a two-dimensional grid. Rows of the grid represent the layers of abstraction or domains, and there are three of them: Problem, Solution, and Implementation. Columns of the grid represent the four pillars of SysML: Requirements, Structure, Behavior, and Parameters. A cell at the intersection of the row and the column represents a view specification to describe what SysML views (diagrams, tables, etc.) and what element types (e.g., blocks, requirements, etc.) must be used at the particular step of system architecture definition.

2) General Workflow

The modeling workflow determined by the MBSE Grid framework is top-to-bottom and left-to-right. That is, the model of the system architecture is built going all the way from the Stakeholder Needs to the Implementation Requirements cell. The whole Problem domain including the black- and the white-box analysis, is about understanding what do the stake-holders expect from the system. Therefore, the Problem domain of the MBSE Grid aligns to the entire Stakeholder Needs and Requirements Definition process defined in the ISO/IEC/IEEE 15288 standard. When the stakeholder needs are clear, the Solution domain can start. The System Requirements cell is the first one to visit. It aligns with the System Requirements Definition process of the ISO/IEC/IEEE 15288 standard. This technical process covers the Subsystem Requirements and Component Requirements cells, too. The rest of system- and subsystem-level cells of the Solution domain align with the Architecture Definition process of the ISO/IEC/IEEE 15288 standard. The same technical process also covers the component-level cells, but only partially. These cells are covered by the Design Definition process, too. This is where the architecture relates to design, as once it is possible to identify the discipline (technical domain) of each component, the Design Definition process can be carried out. Finally, the Implementation process, which transforms requirements, architecture, and design into the system elements using selected implementation technology, is carried out. This process aligns to the Implementation domain of the MBSE Grid framework, but only partially because it includes only the Implementation Requirements cell.

			Pillar		
		Requirements	Structure	Behavior	Parameters
Domain	Problem – Black Box	Stakeholder Needs	System Context	Use Cases	Measures of Effectiveness
	Problem – White Box		Conceptual Subsystems	Functional Analysis	MoEs for Subsystems
	Solution	System Requirements	System Structure	System Behavior	System Parameters
	Solution	Subsystem Requirements	Subsystem Structure	Subsystem Behavior	Subsystem Parameters
	Solution	Component Requirements	Component Structure	Component Behavior	Component Parameters
	Implementation	Implementation Requirements			

Fig. 1. Layout of MBSE Grid Framework

B. Overview

The meta-model consists of four major parts: (i) Problem domain Black-Box meta-model; (ii) Problem domain White-Box meta-model; (iii) Solution domain meta-model; (iv) Traceability relation-ships. The Implementation domain meta-model is not included. Being closely coupled to the specific toolset and less methodology and language dependent, this domain was excluded from our research in order to reduce misperception and ambiguity.

To specify the meta-model, UML was used: all the meta-model elements (artefacts) are captured as UML Classes and relationships among them as UML Associations. Each association has name, direction, and multiplicities on both ends. Each artefact has a mapping to the relevant SysML 1.6 stereotype.

C. Meta-Model Details

Further in the paper, MBSE Grid meta-model concepts are described in detail and mapped to the modeling language concepts.

1) Description of Problem Domain: Black-Box Meta-Model

The goal of the Problem domain analysis phase from the Black-Box perspective is to understand how the SoI interacts with its environment and how it satisfies stakeholder needs. In this phase, the SoI is considered a black box and analysis of it is not concerned with its internal structure and behavior.

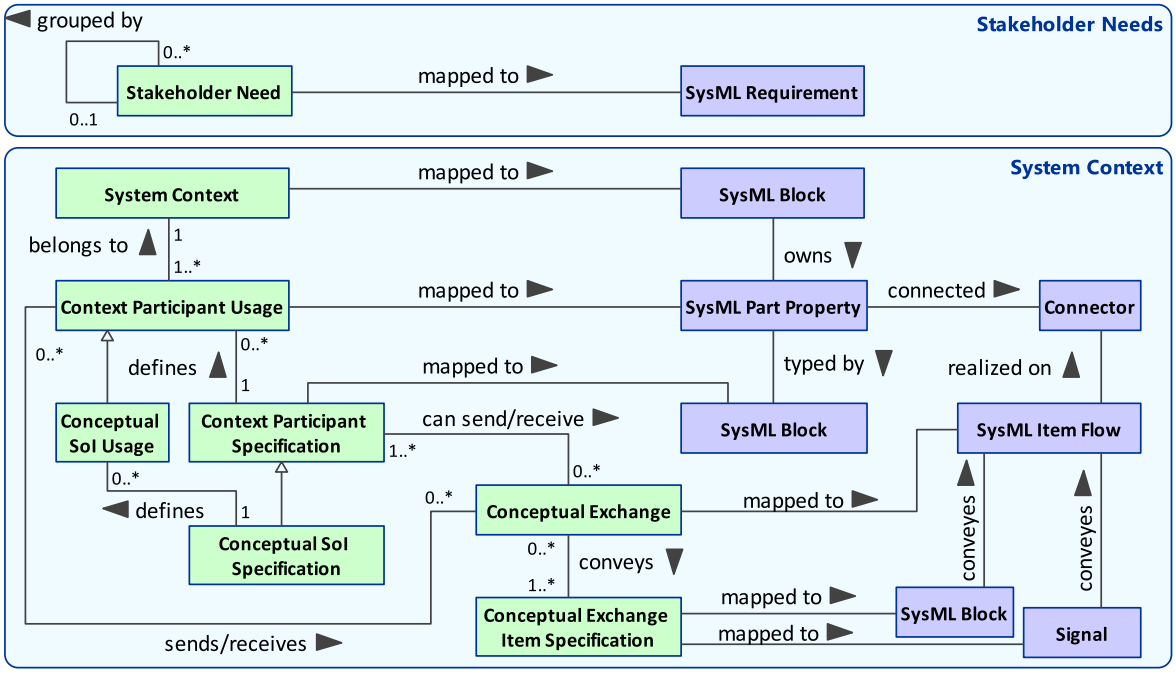

Fig. 2. Meta-model of the Stakeholders Needs and System Context cells (engineering item - ; modeling item -)

TABLE I. META-MODEL ARTIFACTS OF STAKEHOLDER NEEDS AND SYSTEM CONTEXT CELLS

Artifact	Definition	SysML Stereotype
Conceptual Exchange	Flow of information, material, etc. that happens between *Context Participants Usage* elements.	Item Flow
Conceptual Exchange Item Specification	Reusable element for detailing information, material, etc. that flows with *Conceptual Exchanges*.	Block; Signal
Conceptual SoI Usage	Represents *Conceptual SoI Specification* in the specific *System Context*.	Part Property
Conceptual SoI Specification	Definition of SoI at the highest level of abstraction.	Block
Context Participant Usage	Can describe an external element or *Conceptual SoI Usage* and is bound to a specific *System Context*.	Part Property; Swimlane
Context Participant Specification	Definition of *Context Participant*. Can represent SoI or element external to SoI. It is not bound to specific system context and can be reused across different *System Contexts*.	Block
Stakeholder Need	Element used for describing information from various stakeholders of the SoI. This information includes primary user needs, system-related government regulations, policies, procedures, and internal guidelines, among others.	Requirement
System Context	Element describing the situation for interactions between the *Conceptual SoI Usage* and other *Context Participants Usages*.	Block

In this phase, the main inputs and outputs, black-box functions, and quantifiable characteristics of the SoI in a variety of system contexts, are defined. Fig. 3 and Table II disclose how the concepts used in this phase of the Problem domain analysis can be related to SysML stereotypes.

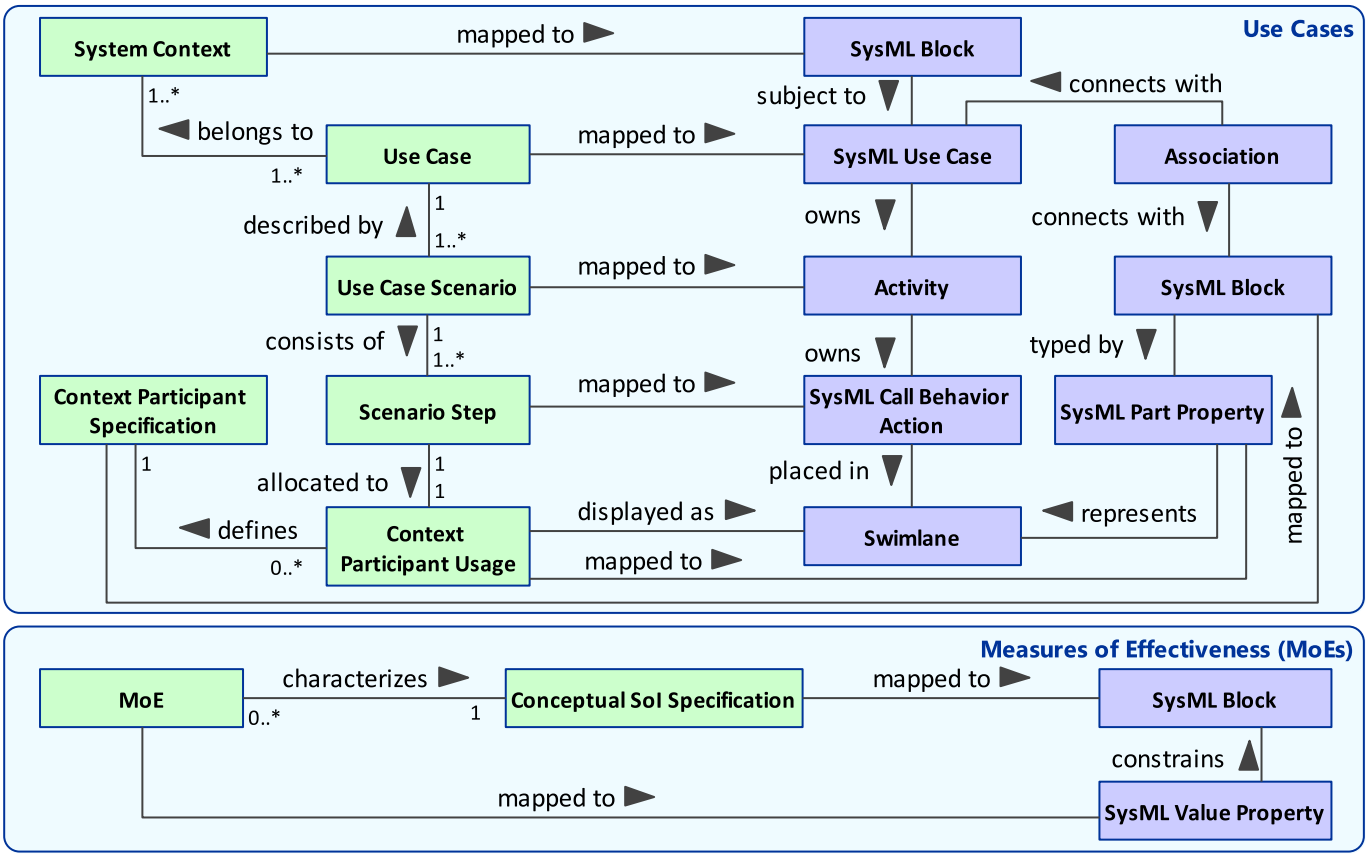

Fig. 3. Meta-model of the Use Cases and Measure of Effectiveness cells
(engineering item - ; modeling item -)

TABLE II. META-MODEL ARTIFACTS OF USE CASES AND MEASURES OF EFFECTIVENESS CELLS

Artifact	Definition	SysML Stereotype
Context Participant Usage	Can describe an external element or *Conceptual SoI Usage* and is bound to a specific *System Context*.	Part Property; Swimlane
Measure of Effectiveness (MoE)	An assessable characteristic of *Conceptual SoI Specification* that refines the particular non-functional *Stakeholder Need*.	Value Property
Scenario Step	An action of the particular *Use Case Scenario*.	Call Behavior Action
Use Case	Element for specifying the expected behavior of the *Conceptual SoI Usage* in the specific *System Context*.	Use Case
Use Case Scenario	A sequence of *Scenario Steps* to define the expected behavior of the *Conceptual SoI Usage*. A single *Use Case* can have one or more *Use Case Scenarios*.	Activity

2) Description of Problem Domain: White-Box Meta-Model

The Problem domain analysis phase from the White-Box perspective is used to understand the expected behavior of the SoI before producing the logical architecture in the Solution domain. This phase should assist in performing more elaborated description of required system functions that can help to identify conceptual subsystems for the SoI. Definition of conceptual subsystems leads to the need of interfaces identification. Plus, every subsystem could have its own quantitative characteristics that need to be identified. This task is done with the introduction of MoE elements for conceptual subsystems. Figs. 4, 5 and Table III include the main elements of this domain and how they can be mapped to SysML stereotypes.

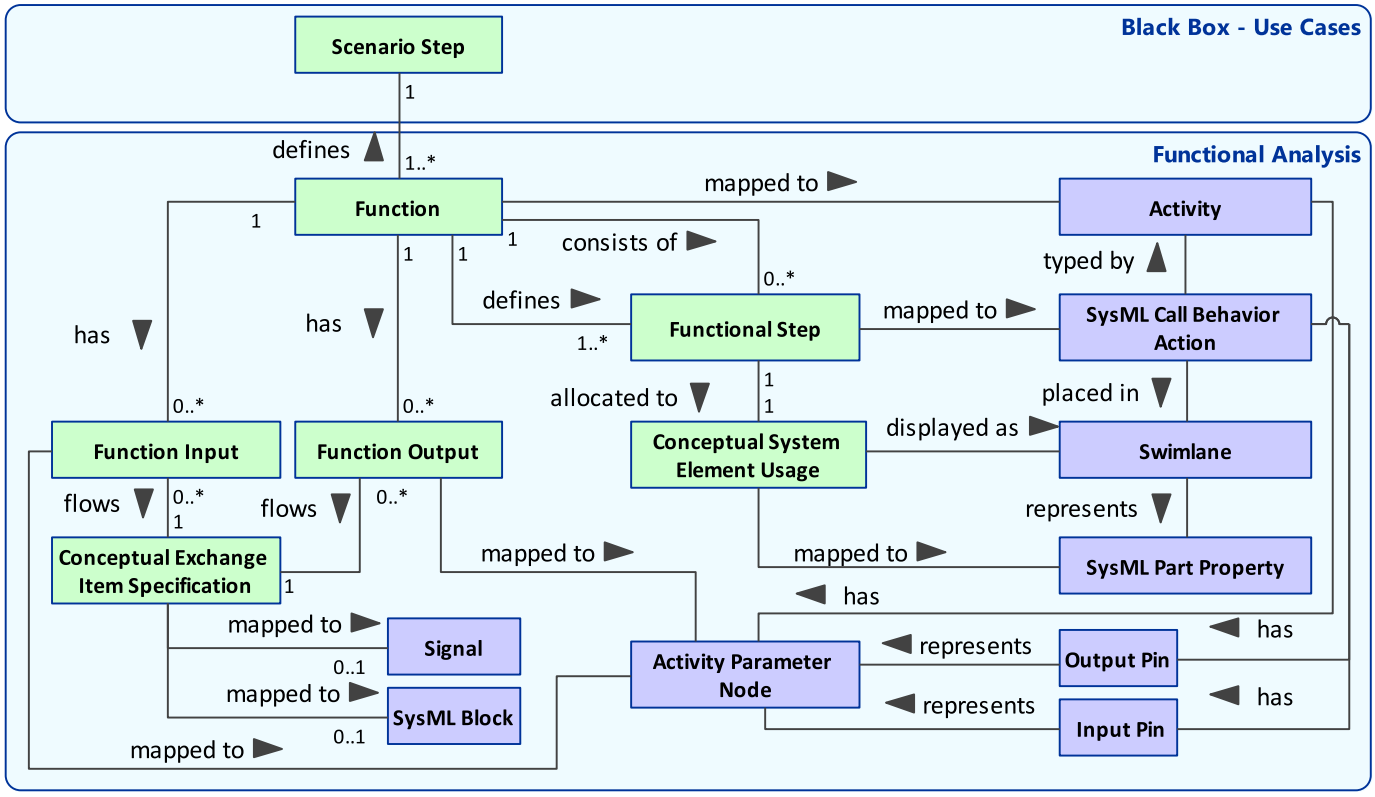

Fig. 4. Meta-model of the Functional Analysis cell (engineering item - ; modeling item -)

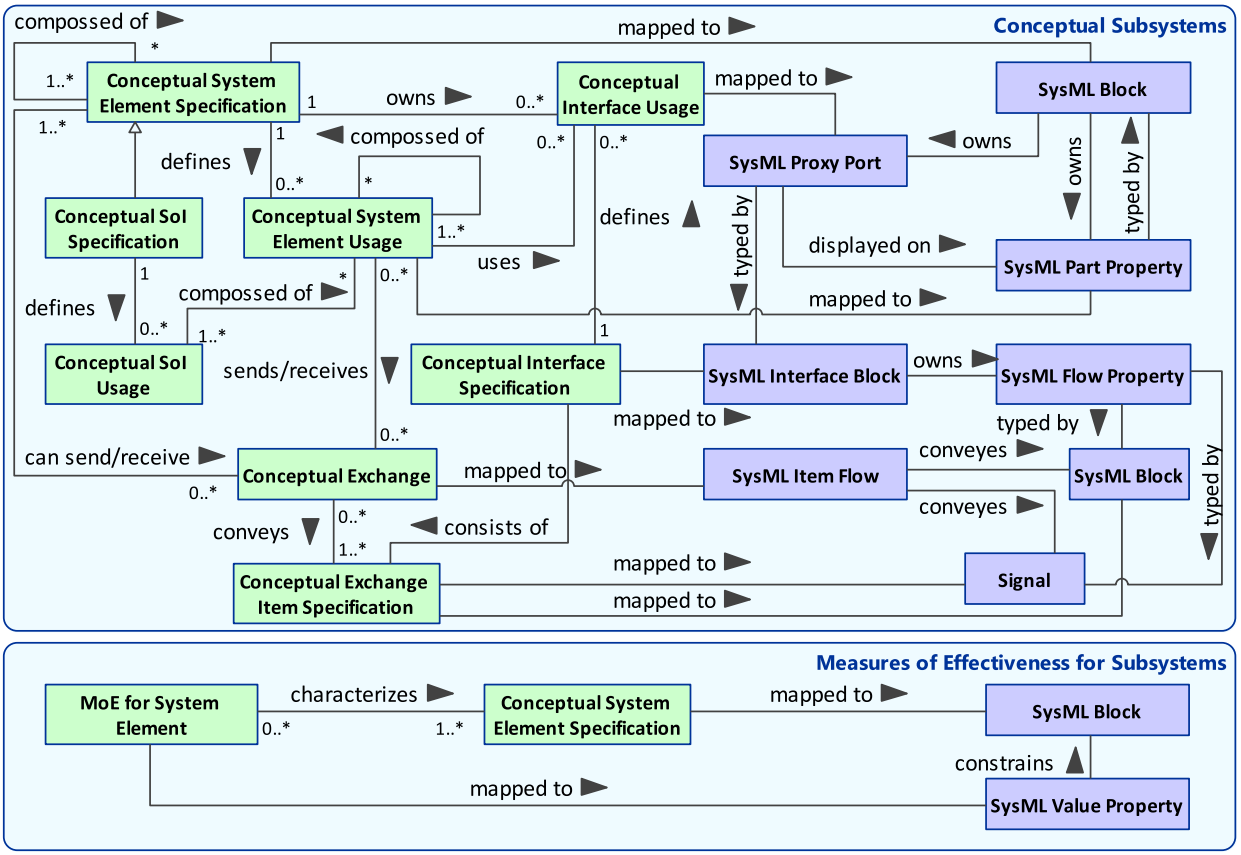

Fig. 5. Meta-model of the Conceptual Subsystems and MoEs for Subsystems cells
(engineering item - ; modeling item -)

TABLE III. META-MODEL ARTIFACTS OF FUNCTIONAL ANALYSIS, CONCEPTUAL SUBSYSTEMS, AND MOES FOR SUBSYSTEMS CELLS

Artifact	Definition	SysML Stereotype
Conceptual System Element Usage	An element of *Conceptual SoI Usage*. Same characteristics are applied as for *Conceptual SoI Usage*.	Part Property; Swimlane
Function	An activity that is performed by the *Conceptual SoI Usage* or its internal elements and consumes and produces *Conceptual Exchanges* set of inputs into a set of outputs.	Activity
Function Input	A single input into the function. Can represent data, material, information, etc.	Activity Parameter Node
Functional Output	A single output of the function. Can represent data, material, information, etc.	Activity Parameter Node
Functional Step	An action of the function. Sequence of functional steps defines a single function.	Call Behavior Action
Conceptual Interface Specification	A declaration of allowed incoming and/or outgoing *Conceptual Exchange Item Specifications*, which governs a single point of interaction.	Interface Block; Flow Property
Conceptual Interface Usage	A declaration of allowed incoming and/or outgoing *Conceptual Exchange Item Specifications*, which governs a single point of interaction in the *System Context* or *Conceptual System Element Specification*.	
Conceptual System Element Specification	Definition of a system element at the conceptual level of abstraction. Can also represent *Conceptual SoI Specification*.	Block
MoE for System Element	An assessable characteristic of *Conceptual* or *Logical System Element Specification*. In Problem domain, it refines a non-functional *Stakeholder Need*. In Solution domain, it is used to satisfy a particular *System/Subsystem Requirement*.	Value Property

3) Description of Solution Domain Meta-Model

The solution domain model defines a precise cross-discipline logical architecture of the SoI. This domain model consists of requirements specification, structure, behavior, and parameters that describe SoI. From MBSE Grid perspective, logical architecture definition process is exactly the same way at system, sub-system, component, or any other logical architecture level. For this reason, in this paper we are not describing how subsystem, component, or other level concepts map to meta-model elements and SysML; we focus only on logical architecture mapping at the a system level. Fig. 6 and Table IV display solution domain meta-model elements and their mappings to SysML stereotypes.

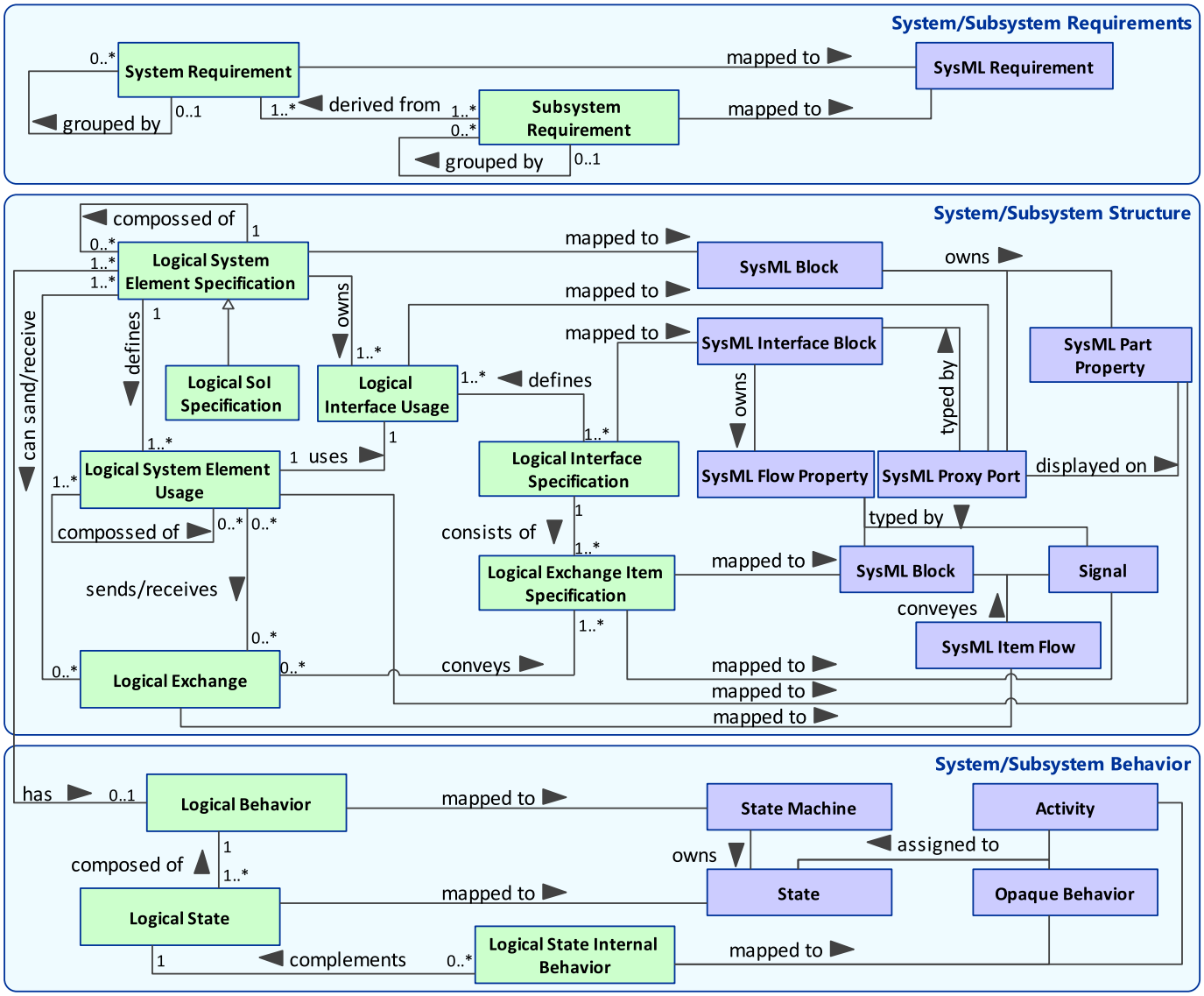

Fig. 6. Meta-model of the System Requirements, System Structure, and System Behavior cells
(engineering item - ; modeling item -)

TABLE IV. META-MODEL ARTIFACTS OF SYSTEM REQUIREMENTS, SYSTEM STRUCTURE, AND SYSTEM BEHAVIOR CELLS

Artifact	Definition	SysML Stereotype
Logical Behavior	The effect produced when an instance of a complex logical system is used in its operational environment. (Created for SEBoK)	State Machine
Logical System Element Usage	An inner element of *Logical SoI Usage*. Same abstraction characteristics should be applied as for *Logical SoI Usage*.	Part Property
Logical System Element Specification	Reusable definition of *Logical System Element Usage*.	Block
Logical Exchange	Element that is used for describing information/data/material/etc flows between *Logical System Element Usages* and *Logical SoI Usage*.	Item Flow
Logical Exchange Item Specification	Specification of flow that is used as *Logical Exchange* between *Logical Systems Elements Usages* and *Logical SoI Usage*.	Block; Signal
Logical Interface Specification	A declaration of allowed incoming and/or outgoing *Logical Exchange Item Specifications*, which governs a single point of interaction.	Interface Block; Flow Property
Logical Interface Usage	A declaration of allowed incoming and/or outgoing *Logical Exchange Item Specifications*, which governs a single point of interaction for *Logical System Element Specification*.	Proxy Port
Logical SoI Specification	The SoI specification viewed from the logical architecture	Block

Artifact	Definition	SysML Stereotype
	perspective. Defines SoI in more details compared to *Conceptual SoI Specification.*	
Logical State	A possible mode in which *Logical SoI Usage* or *Logical System Element Usages* can reside during its lifecycle.	State
Logical State Internal Behavior	Behavior that can be triggered when *Logical SoI Usage* or *Logical System Element Usages* enters a specific *Logical State.*	Activity; Opaque Behavior
Subsystem Requirement	Formal statement that identifies a functional or non-functional characteristic that is needed by a logical subsystem in order to fulfill stakeholders' needs.	Requirement
System Requirement	Formal statement that identifies a functional or non-functional characteristic that is needed by a logical system in order to fulfill stakeholders' needs.	Requirement

Solution domain system model may define architectures of several alternative solutions. For this reason, solution domain allows to define objective functions that can support trade study/trade-off analysis to choose the optimal solution alternative. Fig. 7 and Table V display these elements from the MBSE Grid perspective and their mappings to SysML stereotypes.

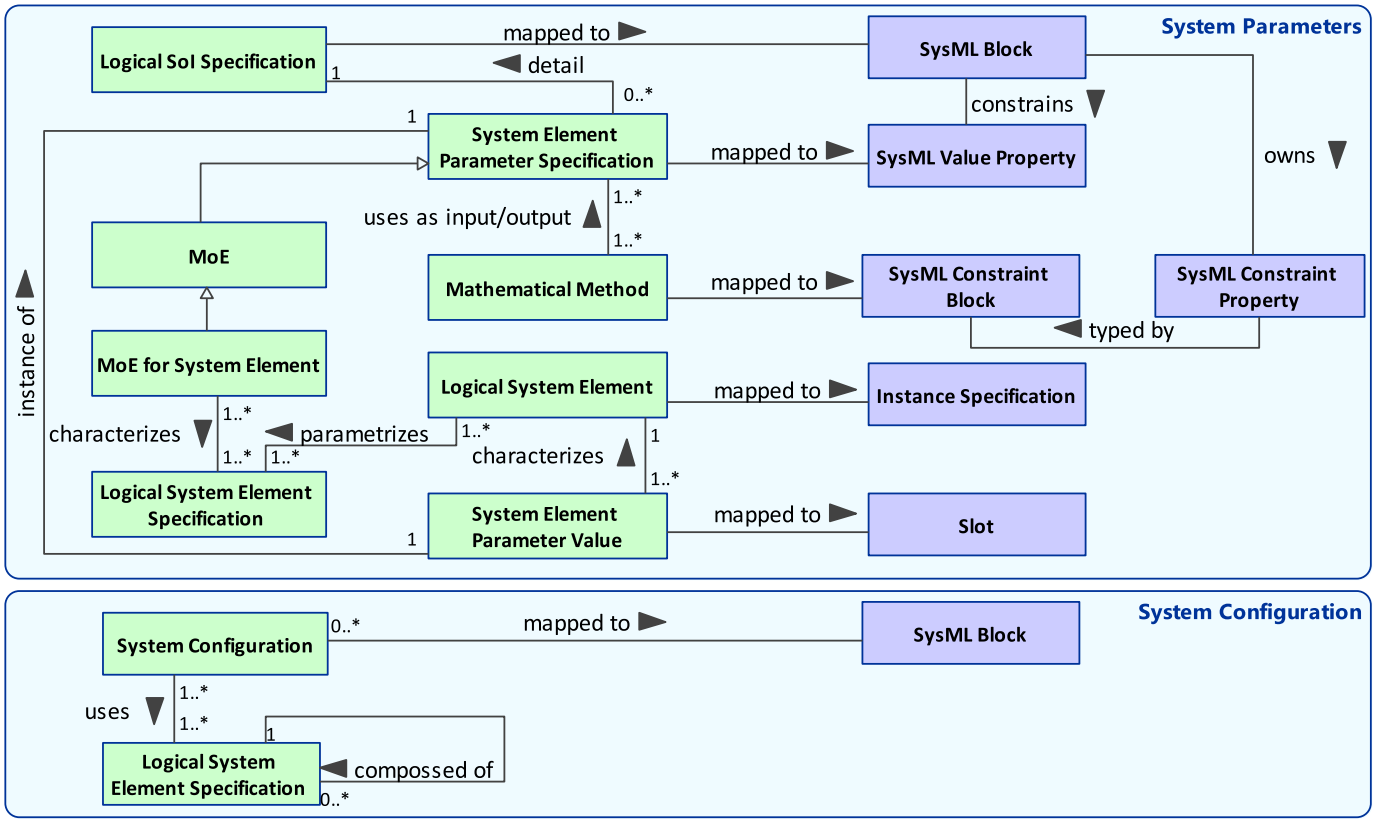

Fig. 7. Meta-model of the System Parameters and System Structure (of the system configuration) cells (engineering item - ; modeling item -)

TABLE V. META-MODEL ARTIFACTS OF SYSTEM PARAMETERS AND SYSTEM STRUCTURE (OF THE SYSTEM CONFIGURATION) CELLS

Artifact	Definition	SysML Stereotype
Logical System Element	An instance of the *Logical System Element Specification*.	Instance Specification
Mathematical Method	Symbolic expression that is used for deriving value of *System Parameter*.	Constraint Block; Constraint Property
System Configuration	Container that encapsulates possible *Logical SoI* architectures with their system elements and allows objectively select the best architecture variant.	Block
System Element Parameter Specification	Property of *Logical SoI* or *Logical System Element* that can be measured with some sort of value.	Value Property
System Element Parameter Value	An actual value of the *System Element Parameter Specification*.	Slot

4) Description of Traceability Relationships Meta-Model

Traceability (or cross-cutting) relationships can be horizontal, vertical, or diagonal. Horizontal traceability relationships are created from the elements that describe conceptual system architecture (in the Problem domain model) or logical system architecture (in the Solution domain model) to the requirements of the relevant domain. These are required to prove that elements describing SoI cover relevant requirements. Horizontal traceability relationships can also be created from the elements that capture the system behavior (as-expected or to-be) of the SoI to the elements that capture its internal structure. Vertical traceability relationships are established between the elements of the same pillar in different domains, for example, from the system requirements to stakeholder needs or from the el-ements of the logical system structure to the elements of the conceptual system structure. Diagonal traceability relationships are created from any system requirement to any element of the Problem domain to assert that this system requirement appears in the specification because of that element.

Fig. 8 and Table VI depicts all traceability relationships between different concepts of MBSE Grid in the meta-model. Also, reveals what SysML relationships they map to.

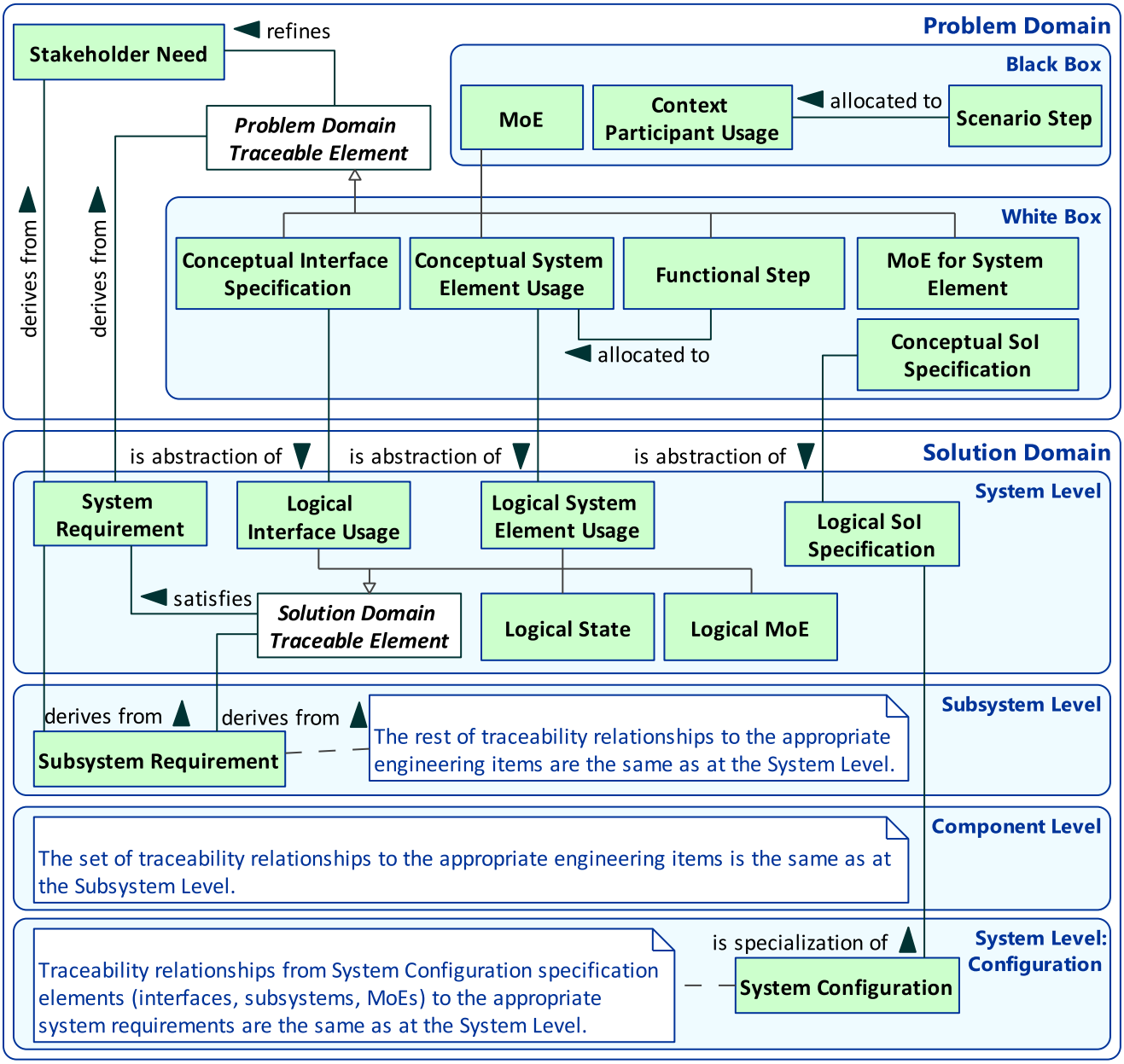

Fig. 8. Meta-model of traceability relationships

TABLE VI. META-MODEL OF TRACEABILITY RELATIONSHIPS

Artifact	Definition	SysML Stereotype
allocated to	Relates system *Functions* to *Conceptual System Elements* when there is a need to trace performing *Function* to *Conceptual System Element* structure.	Allocation
derived from	Relates more abstract requirement to more concrete and is used when there is a need to demonstrate different abstraction level requirements traceability.	DerivedReqt
derives from	Relates any problem or solution domain system model element to *System/Subsystem Requirement* when there is a need to demonstrate how that requirement was derived from more abstract system definition element.	Refine
is abstraction of	Relates any *Conceptual System Element Specification* element to any *Logical System Element Specification* element when there is a need to show that higher abstraction level element abstracts lower abstraction level element.	Abstraction
is specialization of	Relation that allows to demonstrate how lower abstraction level system definition inherits features from higher abstraction level system definition.	Generalization
refines	Relates any problem domain element to one or more *Stakeholder Needs* when there is a need to refine *Stakeholder Needs* with *Conceptual System Elements*.	Refine
satisfies	Relates solution domain system structure element to system/subsystem/component requirement when there is a need to demonstrate that system model element satisfies that abstraction level requirement.	Satisfy

IV. MAPPINGS FROM SAMPLE MODEL TO META-MODEL

To illustrate how SysML model elements created by applying the MBSE Grid framework, are mapped to the proposed meta-model elements, the following matrix has been created. While its columns display the meta-model elements, its rows display the elements that belong to the Problem Domain Black-Box model of the Vehicle Climate Control Unit (VCCU), the conceptual SoI. According to the stakeholder needs there is but one system context, the Vehicle In Use, where the SoI interacts with other context participants. E.g., Vehicle Occupant, Temperature Sensor. These interactions are specified as conceptual exchanges captured in the model as item flows conveying the User Command, Ambient Temperature, etc. The Provide Comfortable Temperature use case goes together with scenario, which includes a list of steps, such as Check System, Measure Temperature, and so on. Stakeholder needs also reveal that VCCU has a set of MoEs, that is, soundLevelVCCU, totalEnergyConsVCCU, etc. Fig. 9 is just a small exerpt from the matrix that displays all the mappings including the Problem Domain White-Box and Solution Domain.

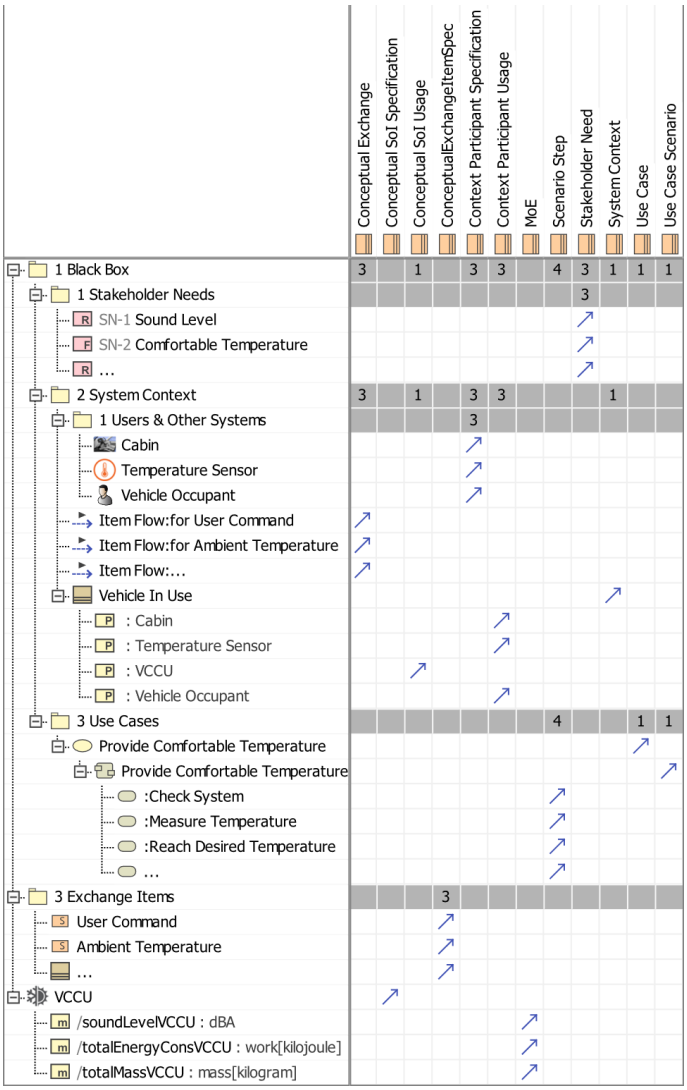

Fig. 9. Mappings from Problem Domain model elements to meta-model

V. CONCLUSIONS AND FUTURE WORKS

The study of existing systems engineering meta-models revealed the terminology gap between the systems architecture frameworks and systems engineering processes. To fill this gap in, we have proposed a systems engineering meta-model based on the terminology used in the MBSE Grid framework and *Systems Engineering Handbook V4*. The proposed meta-model serves as ontology for systems engineers. It allows to use pure SysML to model systems by providing a clear mapping of SysML stereotypes to engineering concepts. Using pure SysML helps to avoid costs of maintaining language extensions. The proposed meta-model is a supplement to the MBSE Grid framework, formalizing its terminology in the form of UML-based meta-model.

The proposed meta-model can serve as a foundation to introduce company-specific terminology. In such case it can become a basis for creation of a company-specific UML profile.

To cover the complete scope of MBSE Grid, the meta-model needs to be extended to include safety and verification and validation concepts.

References

[1] Anonymous

[2] Friedenthal, S, Moore, A, Steiner, R 2014, A Practical Guide to SysML, 3rd Edition, Morgan Kaufmann, Waltham, MA (US).

[3] GfSE (German Chapter of INCOSE) 2021, System Architecture Framework (SAF), November 2021, viewed 9 December 2022 <https://www.gfse.de/arbeitsgruppen/119-kacheln/arbeitsgruppen/arbeitsgruppen-neu/780-system-architecture-framework-saf.html>.

[4] IDEAS Group 2010, DoDAF Formal Ontology, August 2010, viewed 9 December 2022, <https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_ontology1/>.

[5] INCOSE 2015, Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition, INCOSE, San Diego, CA (US).

[6] ISO/IEC JTC 1/SC 7 2015, Systems and software engineering — System life cycle processes, ISO/IEC/IEEE 15288:2015.

[7] ISO/IEC JTC 1/SC 7 2022, Software, systems and enterprise — Architecture description, ISO/IEC/IEEE 42010:2022.

[8] Anonymous

[9] Anonymous

[10] Anonymous

[11] NATO 2018, NATO Architecture Framework (NAF), Version 4, January 2018, viewed 9 December 2022, <https://www.nato.int/nato_static_fl2014/assets/pdf/2021/1/pdf/NAFv4_2020.09.pdf>.

[12] SE^2 Challenge Team 2015, SE^2's Cookbook, viewed 9 December 2022,<https://www.gfse.de/telescope/cookbook-telescope.html>

[13] Silingas, D, Vitiutinas, R, Armonas, A, Nemuraite, L 2009, 'Domain-Specific Modeling Environment Based on UML Profiles', Conference proceedings of the 15th International Conference on Information and Software Technologies, Kaunas, pp. 167-177.

[14] OMG 2022, OMG Unified Architecture Framework (UAF), Version 1.2, July 2022, viewed 9 December 2022, <https://www.omg.org/spec/UAF/1.2>.

[15] OMG 2019, OMG Systems Modeling Language (OMG SysML ™), Version 1.6, November 2019, viewed 9 December 2022, <https://www.omg.org/spec/SysML/1.6/>.

[16] OMG 2017, Unified Modeling Language® (OMG UML®), Version 2.5.1, December 2017, viewed 9 December 2022, <https://www.omg.org/spec/UML/2.5.1/PDF>.

[17] U.S. Department of Defense 2010, DoD Architecture Framework (DoDAF), Version 2.02, August 2010, viewed 9 December 2022, <https://dodcio.defense.gov/library/dod-architecture-framework/>.