Design Ontology Supporting Model-Based Systems Engineering Formalisms

Jinzhi Lu, *Member, IEEE*, Junda Ma, Xiaochen Zheng, Guoxin Wang, Han Li, and Dimitris Kiritsis

***Abstract*—Model-based systems engineering (MBSE) provides an important capability for managing the complexities of system development. MBSE empowers the formalism of system architectures for supporting model-based requirement elicitation, specification, design, development, testing, fielding, etc. However, the modeling languages and techniques are heterogeneous, even within the same enterprise system, which leads to difficulties for data interoperability. The discrepancies among data structures and language syntaxes make information exchange among MBSE models more difficult, resulting in considerable information deviations when connecting data flows across the enterprise. Therefore, this article presents an ontology based upon *graphs*, *objects*, *points*, *properties*, *roles*, and *relationships* with *extensions* (GOPPRRE), providing metamodels that support the various MBSE formalisms across lifecycle stages. In particular, knowledge graph models are developed to support unified model representations to further implement ontological data integration based on GOPPRRE throughout the entire lifecycle. The applicability of the MBSE formalism is verified using quantitative and qualitative approaches. Moreover, the GOPPRRE ontologies are used to create the MBSE formalisms in a domain-specific modeling tool, *MetaGraph*, for evaluating its availability. The results demonstrate that the proposed ontology supports the formal structures and descriptive logic of the systems engineering lifecycle.**

***Index Terms*—Formalism, interoperability, knowledge graph, model-based systems engineering, ontology.**

I. Introduction

THE increasing complexity of technological innovations and their interoperability requirements within systems-of-systems, systems, subsystems, and components have led to overcomplexity of architectures and data structures. In order to manage the system complexity, to design interfaces across systems, subsystems, and components and to improve understandings of the system nature among different stakeholders, model-based systems engineering (MBSE) has been proposed to make use of models for formalizing end-to-end systems engineering perspectives. Each interface between the lifecycle phases poses communication challenges owing to this increasing complexity [1]. Much of the complexity is the result of individual stakeholder interests; they may have different concerns about the systems and artifacts of interest, and they may, in turn, demand unique informational and data-standard feedback. These results can often be seen within the architectural models, as discrepancies among such models create a system-integration issue, resulting in barriers to communications, understandability, and more importantly, operations. Apart from stakeholder nuances, the integration of model views is also challenged by different domain-specific knowledge base and systems engineering taxonomies.

Across the entire lifecycle, enterprise data integration is the ultimate goal for a fully implemented MBSE. However, at the working levels, domain engineers commonly formalize their various domain problems using stove-piped domain-specific modeling (DSM) languages and specifications. These various representations are difficult to piece together during collaborative development, and the results often lead to misinterpreted or inaccurate reporting. For example, Systems Modeling Language (SysML), Business Process Modeling Notation (BPMN), and data flow architecture (DFA) are integrated to formalize the integrated product service architectures, where the BPMN is expected to support process-based service modeling, and the DFA is used to support the data flow architecture design [2]. Therefore, the information representations and data structures should be standardized to provide a complete model flow across the lifecycle while meeting all stakeholder and engineering requirements.

System development is an iterative process that relies on a unified and authoritative data architecture built upon collaboration. Owing to advances in Artificial Intelligence (AI) and machine-learning (ML) techniques, the concept of MBSE is undergoing a digital transformation that will ultimately lead to advanced facilitation to complex system development [3]. Semantics web-data exchange is the basis for much of the current data and information integration via AI reasoning. Thus, participants of the MBSE lifecycle should work to ensure completeness and consistency of the data that fuel decision making and engineering task implementation. Based upon an AI-driven data exchange, the knowledge management of the future aims to provide the required information to stakeholders whenever (or even before) they need it [4].

This article focuses on a unified MBSE ontology based on meta–meta models consisting of six key concepts with extensions: *Graph*, *Object*, *Point*, *Property*, *Role*, and *Relationship*

Manuscript received 7 May 2021; revised 9 July 2021; accepted 16 August 2021. Date of publication 9 September 2021; date of current version 9 December 2022. This work was supported in part by the CN National Key Research and Development Plan under Grant 2020YFB1708100, in part by European Union (EU) H2020 under Project 825030 QU4LITY Digital Reality in Zero Defect Manufacturing, and in part by InnoSwiss IMPULSE project on Digital Twins. *(Corresponding author: Guoxin Wang.)*

Jinzhi Lu, Xiaochen Zheng, and Dimitris Kiritsis are with the SCI STI DK Group, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail: jinzhl@kth.se; xiaochen.zheng@epfl.ch; dimitris.kiritsis@epfl.ch).

Junda Ma, Guoxin Wang, and Han Li are with the Beijing Institute of Technology, Beijing 100081, China (e-mail: mjd2015@sina.cn; wangguoxin@bit.edu.cn; 3120190349@bit.edu.cn).

Digital Object Identifier 10.1109/JSYST.2021.3106195

(GOPPRRE). This ontology presents a formalization solution for the MBSE modeling with a unified syntax and data structure to support systems engineering information exchange via the integration of AI and ML. The main contributions of this defined ontology are as follows:

1) it supports integrated architectural representation across the lifecycle;
2) it promotes MSBE tools built upon data interoperability and consistency;
3) it provides potential solutions for developing AI/ML MBSE roadmaps.

To promote the scalability of the proposed ontology, the ontology will be discussed in the Industrial Ontologies Foundry Systems Engineering Working Group.[1]

The rest of this article is organized as follows. We discuss related works and the proposed research methodology in Section II. In Section III, the designed ontology is analyzed. A case study is presented in Section IV for the evaluation of our ontology using quantitative and qualitative approaches. Finally, Section V concludes this article.

II. Research Design

A. Literature Review

MBSE, which supports complex systems engineering and development efforts [5] by formalizing development processes, system architectures, and operational interrelationships, has been widely applied in different industrial sectors. There are currently several modeling languages in use (e.g., SysML [6], object process methodology [7], and BPMN [8]), which provide modeling tools to describe real-world artifacts and processes using graphical views. Recently, researchers have proposed an Object Management Group standard for model-driven engineering, comprising a four-layer architecture [9]. The four layers are labeled as M0–M3 and provide the modeling framework needed to support MBSE. The M0–M3 layers are described thoroughly in Section III of this article. In order to formalize the MBSE formalisms, the meta–meta models GOPPRR formalizations were applied for specific system view representations [10]. Several generic modeling environments also provided metamodeling languages that can support the complex system development based on unified modeling-language (UML) notation and object constraint language [11]. Advocates of these methods continue to seek a singular adaptive language that can be used to describe all system architecture views.

In addition to implementing MBSE practices using one specific language, some researchers also proposed MBSE solutions based on the integration of multiple modeling languages to formalize the target systems for a unified view with different domain features. For example, SysML and BPMN were integrated to support manufacturing information acquisition system formalisms [12]. Specifically, the BPMN was used to describe the process configurations for different clients. When developing the vehicle architectures, SysML models were used to represent the physical structure that was integrated with EAST-ADL (a domain-specific language for vehicle architecture development) [13]. The UML and BPMN were integrated to construct the unified representation of an enterprise architecture [14]. The BPMN was focusing on the business process description and the UML was used to represent the architectural views of the enterprise. Moreover, some researchers integrated system languages with simulation languages for supporting automated verification. The SysML-Modelica transformation specification was used to transform SysML architecture models to Modelica models for automated simulations [15].

Through the integrated features of these languages, different views of the enterprise architecture are presented in one MBSE tool. Current MBSE languages have their own features to support specific purposes. However, when applying MBSE to support the complex system development, each domain knowledge requires MBSE modelers developing their own solutions based on different languages. Though supported by a few MBSE tools, the integration of different languages is still challenging, especially when a unified formalism is required for representing the domain-specific knowledge in the different MBSE tools.

Currently, middleware is widely used to facilitate the information exchange between different MBSE tools. For example,, extensible XML Metadata Interchange (XMI) is used to support the data exchange between SysMLs and multiple other tools [16]. Moreover, ontological methods are considered as novel middleware, which are used to cope with lower level tool and data interoperability, consistency issues, and reasoning. For example, MBSE ontologies have been developed to formalize domain-specific concepts and their interrelationships using different languages [17], [18]. Some researchers have provided ontology-based approaches to facilitate the design automation for complex systems [19]. In addition, ontology and formalisms are developed for systems engineering. Yang *et al.* [21] provided a unified ontology to describe a systems engineering body of knowledge for the International Council on Systems Engineering [20]. Seidner and Roux [22] proposed a formal method (ontology concepts) for behavior simulation, which is used for safety analyses when implementing systems engineering. Lu *et al.* [23] developed an ontology to support automated cosimulation using an MBSE tool chain. The ontology was used to implement MBSE models for integrated verification. Most of the aforementioned ontological approaches, however, focused on domain-specific problems instead of modeling languages and data interoperability across the entire MBSE lifecycle.

MBSE and ontology are the basis for constructing a digital replication technology and supporting virtual verification concepts [24], [25]. They are expected to provide potential solutions for combining systems engineering approaches and AI technologies. Some researchers have provided an ontology-based approach facilitating the design automation for complex systems [19], [25]. Hao *et al.* [26] proposed an ontology-based method to support knowledge management. Ontology contributes to semantics descriptions and models that support decision making regarding the system development and real-time operations via a universal system description and information transfer [27], [28].

[1]https://www.industrialontologies.org/

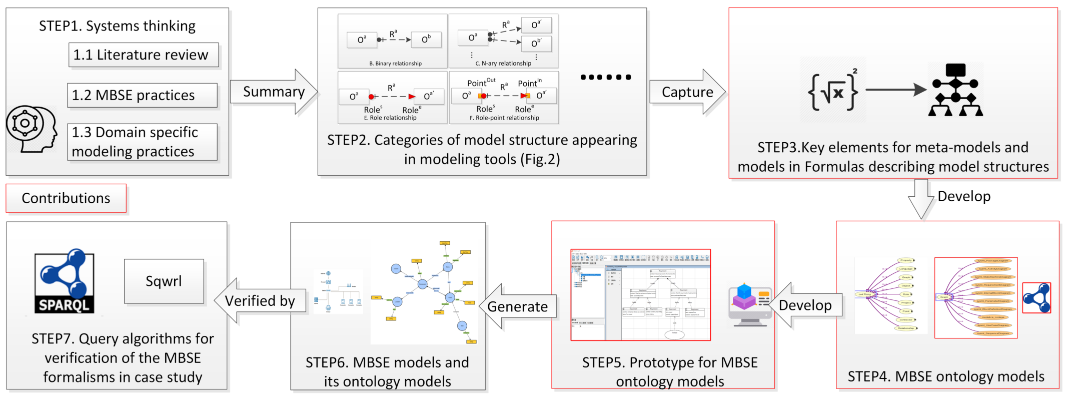

STEP1. Systems thinking
1.1 Literature review
1.2 MBSE practices
1.3 Domain specific modeling practices
Summary
STEP2. Categories of model structure appearing in modeling tools (Fig.2)
Capture
STEP3.Key elements for meta-models and models in Formulas describing model structures
Develop
STEP4. MBSE ontology models
Develop
STEP5. Prototype for MBSE ontology models
Generate
STEP6. MBSE models and its ontology models
Verified by
Sqwrl
SPARQL
STEP7. Query algorithms for verification of the MBSE formalisms in case study
Contributions

Fig. 1. Research methodology.

B. Summary

Several modeling languages have been used to formalize the different views and approaches found in the systems engineering lifecycle. However, many challenges arise when these different languages are adopted for different enterprises.

1) Different modeling languages are used for different purposes of the system development. When formalizing domain-specific problems, different knowledge representations require different modeling languages. Such heterogeneous model structures and syntax result in difficulties for supporting the unified formalism of a specific domain; they do not support multiple system views by integrated representations across modeling languages.
2) During the entire lifecycle, stakeholders often use different MBSE languages for formalizing system architectures and verifying system performances. These languages need to be integrated when combining domain-specific knowledge with system architectural views in order to maintain the consistency of system models, domain models, and verification models.
3) An ontology, which supports MBSE formalisms, enables combining systems engineering processes and AI tools for enterprise knowledge management and decision making.

In this article, Web ontology language (OWL) is used to design a complete MBSE ontology based on the GOPPRRE approach that can support the information exchange across the enterprise.

C. Research Methodology

The research methodology of this article is depicted in Fig. 1.

1) *Steps 1 and 2:* We make the use of a systems thinking approach, including literature review, MBSE practices, and DSM practices to identify the possible model structures that appear in modeling tools.
2) *Step 3:* Based on the available model structures, we make use of the GOPPRRE approach to define the formulas for describing these structures. The formula aims to identify all the metamodel and model elements based on the GOPPRRE approach.
3) *Step 4:* The next step is to develop MBSE ontology models, which are built with Protégé based on the proposed formulas.
4) *Steps 5 and 6:* In order to implement the MBSE ontology in practice, a DSM tool, MetaGraph, is developed. This tool enables to generate MBSE ontology models from MBSE models automatically.
5) *Step 7:* Finally, SPARQL and SQWRL languages are used in a case study to query the ontology models in order to verify if all the model elements are generated from MBSE models in MetaGraph.

III. Ontology Design for MBSE Formalism

In this section, we first identify the problem statement of the MBSE formalisms across the MBSE languages. Then, the GOPPRRE approach is introduced. Finally, knowledge graph models are used to formalize the MBSE models.

A. Problem Statement

During the entire lifecycle, system development, system artifacts, and disciplinary knowledge are the three main aspects that should be considered when developing MBSE models [29], [30]. To demonstrate different views, model components are connected by different reference connections in different MBSE languages. Such languages have different connecting patterns (how the model components connect to each other) that are important to demonstrate the modelers' ideas across the MBSE languages. Thus, when describing the MBSE models, we focus on the ontology definition to identify and describe all the connecting patterns and their related entities in the MBSE formalisms. As shown in Fig. 2, there are six types of connecting patterns, including the following [31].

1) Reference relationship refers to a simple connecting reference between O^a and O^b, which belong to two types of model components: a and b.
2) Binary relationship refers to one connecting reference R^a between only two given components O^a and O^b. The R^a is binary and has its own attributes.
3) N-ary relationship refers to one connecting reference R^a between more than two different components. The R^a is binary and has its own attributes.
4) Set relationship refers to one connecting reference R^a between two sets of components. Each set includes several

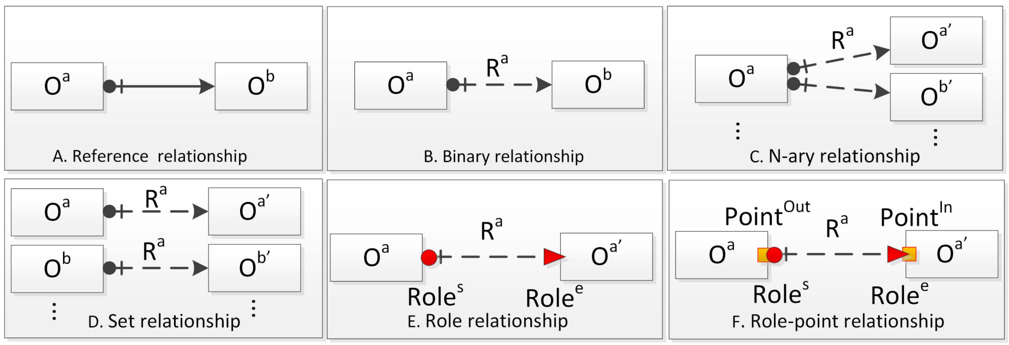

A. Reference relationship
B. Binary relationship
C. N-ary relationship
D. Set relationship
E. Role relationship
F. Role-point relationship
O^a O^b $O^{a'}$ $O^{b'}$ R^a
$Role^s$ $Role^e$
$Point^{Out}$ $Point^{In}$

Fig. 2. Connecting patterns for MBSE formalisms.

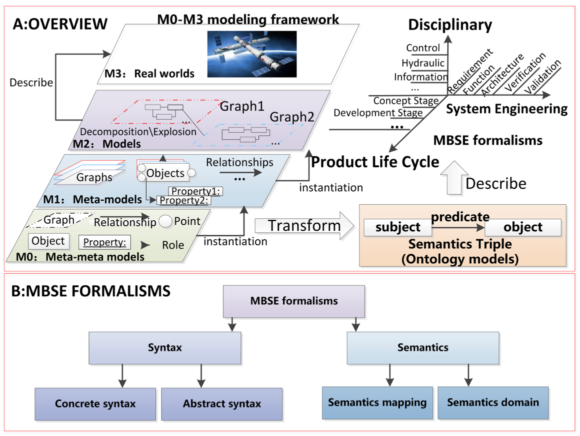


Fig. 3. (A) Overview of the ontology. (B) MBSE formalisms and their compositions.

types of model components. The R^a is binary and has its own attributes.

5) Role relationship refers to one connecting reference R^a with two ends (role concepts) $Role^s$ and $Role^e$, between more than two different components. The role concepts also have their own attributes.

6) Role-point relationship refers to one connecting reference R^a with two ends (role concepts) $Role^s$ and $Role^e$. Moreover, the connecting reference is connected to the components O^a and O^b through their points: $Point^{Out}$ and $Point^{In}$. The point concepts have their own attributes.

To realize the unified formalisms for such connecting patterns, a GOPPRRE approach is proposed.

B. Overview

The overall workflow of the proposed approach is shown in Fig. 3(A). An M0–M3 modeling framework is proposed to develop the MBSE ontology models (semantic triple), including the following.

1) M3: Meta–meta models that refer to basic elements of the constructed model compositions and their interconnections. We adopt GOPPRR meta–meta models and their extensions to support metamodel development. The details of GOPPRR meta–meta models are introduced in the next section.

2) M2: Metamodels refer to the model compositions and connections needed to develop models. Based on meta–meta models, metamodels are constructed to develop models.

3) M1: Using metamodels, the MBSE models are developed to represent real-world systems.

4) M0: Real-world artifacts are considered, including disciplinary, systems engineering perspectives, and development processes of complex systems.

Using the M0–M3 framework with GOPPRR meta–meta models, the ontology is developed based on semantics triples (i.e., subject, object, and predicate) to formalize complex systems from the following three dimensions: disciplinary, system lifecycle, and system engineering perspectives [29]. The disciplinary dimension includes several domains, such as control engineering, mechanical engineering, etc. Systems engineering perspectives refer to requirements, functions, architectures, etc. The product lifecycle includes different processes of the complex system development.

As shown in Fig. 3(B), the MBSE formalisms include syntax and semantics [32]. Syntax refers to the representations of the MBSE formalisms, and semantics refers to the MBSE model meanings. The details are explained as follows:

1) Abstract syntax refers to the compositions of the MBSE models and their defined connecting partners for describing links between each composition. It is realized using the core GOPPRRE concepts for the MSBE formalisms (introduced in Section III-C)

2) Concrete syntax refers to the visual representations of the MBSE model compositions. It is represented in the knowledge graph model as the annotation property, which is introduced in Section III-D.

3) Semantics domain refers to the target of the semantic mapping, which implies the meanings of the MBSE models. It includes the three dimensions shown in Fig. 3(A). The formalisms are used to describe the system artifacts, product–lifecycle processes, and disciplinary knowledge needed to support the information exchange during the system development.

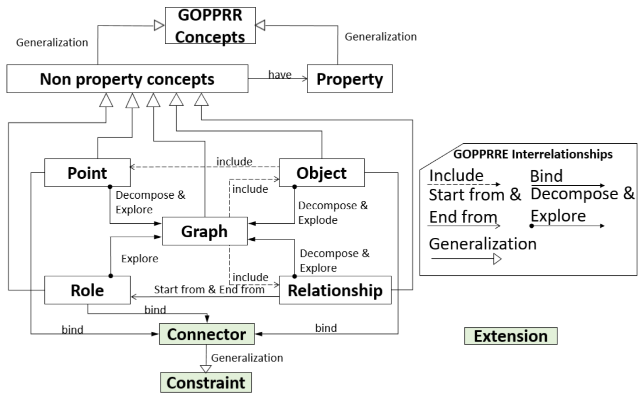


Fig. 4. GOPPRRE meta–meta models, interrelationships and their extensions.

TABLE I
INTERRELATIONSHIPS AMONG GOPPRRE META–META MODELS

GOPPRR	$graph_a^{id}$	$object_b^{id}$	$relationship_d^{id}$	$role_c^{id}$	$point_e^{id}$
$graph_a^{id}$	-	decompose explode	decompose explode	explode	decompose explode
$object_b^{id}$	include	-	-	connect	-
$relationship_d^{id}$	include	-	-	-	-
$role_c^{id}$	-	-	startFrom endTo	-	-
$point_e^{id}$	-	include	-	connect	-
$property_f^{id}$	have	have	have	have	have

4) Semantics mapping refers to the dependencies among the MBSE models and their meanings according to the three dimensions.

C. GOPPRRE Concepts for MBSE Formalisms

The *GOPPRRE* approach uses the M0–M3 modeling framework, as inspired by the GOPPRR meta–meta models and their extensions, to construct the MBSE model syntax and semantics. We added one new concept, *constraint* to define the connecting patterns within abstract syntax when developing metamodels and models. In Fig. 4, the details of the GOPPRRE concepts are introduced, which are as follows.

1) *Graph* is an entity collection of *Object*, *Relationship*, and *Role*, represented in one layout (e.g., a UML class diagram). The graph can be an independent visual diagram.[2] Moreover, a graph can be defined to represent a virtual diagram having decomposition and explosion interrelationships from one *Object* or other elements (as shown in Table I) in its or other *Graph*.
2) *Object* is an entity that constructs a *Graph*, for example, a requirement block in a SysML Requirement diagram.
3) *Point* is one attached port in an *Object*, for example, one port for value property block in a SysML internal block diagram.
4) *Relationship* refers to one connection between the *Points* and/or *Objects*, for example, a satisfy connection in a SysML Requirement diagram.
5) *Role* is used to define the binding restrictions with the relevant *Relationship*. One *Relationship* is associated with two *Roles*, for example, the two end arrows of the satisfy connection in a SysML Requirement diagram. Through each role, the *Relationship* is defined as one that binds with one *Point* or one *Object* in its one end.
6) *Property* is a specific attribute of metamodels that is attached to the other five meta–meta models, including *Graph*, *Object*, *Relationship*, *Role*, and *Point*.
7) *Extension* refers to the additional *constraints* used to construct metamodels. In this article, one constraint is developed as a *connector*. It refers to one binding between one *Point* or *Object* and one *Role* in one side of the *Relationship*.

As shown in Fig. 4, *GOPPRR* concepts are the basic compositions of meta–meta models, including *nonproperty concepts* and *Property*. The *nonproperty* concepts and *Property* have all of the attributes of the *GOPPRR* concepts, such as system ID (the unique identification for each concept) and local label (display in the tool). Extension refers to the extra concepts used for constructing metamodels and models including *constraint* and *connector*. The *connector* is one type of *constraint* with all its attributes, which is used for defining the connecting rules among *Objects* and *Points*.

Each *nonproperty concept*, such as *Object*, is allowed to decompose into one *Graph* or to explode into one or more *Graph*. These are two ways to represent the mappings from each *Object*, *Relationship*, *Role* and *Point* to the new *Graphs*. When building models, new models can be linked to these *nonproperty concepts* to represent the interrelationships of decomposition and explosion among them as follows:

1) Decomposition, referring to a breakdown mapping from *nonproperty concept* to one new *Graph*. When developing a new *Graph* model, it represents a breakdown system structure of the related *nonproperty concept*.
2) Explosion, referring to mappings from one *nonproperty concept* to one or more *Graph*. When developing a new *Graph* model, it represents additional architectural views of the related *nonproperty concepts*.

Definition 1: Token ::= refers to a collection of elements. Token $\vee$ refers to a union of elements. As shown in Table I, the GOPPRRE meta–meta models are identified, and their interrelationships are defined.[3] Thus, the metamodel, *Graph*, is defined as

$$\text{graph}_{Tp} ::= \Big\{ \sum \text{object}^{obTp}, \sum \text{relationship}^{reTp} \\ \sum \text{role}^{roTp}_{reTp}, \sum \text{point}^{poTp}_{obTp}, \sum \text{property}^{\text{pro}Tp}_{\text{nonPro}}) \Big\} \quad (1)$$

where graph_{Tp} refers to the ontological concept of a metamodel, *Graph*, whose type is defined as Tp; object^{obTp} refers to the ontological concept of the metamodel, *Object*, where $obTp$ is a type of *Object*; The $\text{relationship}^{reTp}$ refers to the ontological concept of metamodel *Relationship*, where

[2]A diagram refers to a symbolic representation of model topology (modeling windows for each model) in a modeling tool such as Fig. 6(A).

[3]Read as $\text{graph}^{\text{id}}_a$ include $\text{object}^{\text{id}}_b$.

$reTp$ is a type of *Relationship*; $\text{role}_{reTp}^{roTp}$ refers to the ontology concept of a metamodel, *Role*, and $reTp$ refers to the *Relationship* that starts from (ends at) the *Role*, whose type is $roTp$; $\text{point}_{obTp}^{poTp}$ refers to the ontological concept of the metamodel, *Point*, and $obTp$ refers to the *Object*, including the *Point*, whose type is $poTp$; and $\text{property}_{\text{non}Pro}^{\text{pro}Tp}$ refers to ontological concept of the metamodel, *Property*, and nonPro refers to the *nonproperty* elements (*nonproperty* $\subseteq \{Graph, Object, Relationship, Role, \text{and } Point\}$), having the *Property* of type, $proTp$. This formula represents that a metamodel *Graph* is defined with associated metamodels of *Object*, *Relationship*, *Role*, *Point*, and all their *Property*.

To define the connection rules among metamodels *Objects* and *Points* in each *Graph*, an additional constraint is defined as a *connector* as follows:

$$\text{connector}(\text{conId}) ::= \Big\{\text{relationship}^{reTp}, \text{role}_{reTp}^{roTp} \\ \text{object}^{obTp}(\vee \text{point}_{obTp}^{poTp})\Big\} \tag{2}$$

where the connector(conId) defines a rule conId that allows metamodel *Relationship* $reTp$, metamodel *Role* $roTp$, or metamodel *Object* *obTp* (or metamodel *Point* *poTp* in *obTp*) to be connected. Through this definition, one side of the *Relationship* $reTp$ is binding to one *Object* or *Point* through *Role*.

$$\text{graph}_{Tp}(g\text{Id}) :: \\ = \Big\{\sum \text{object}^{obTp}(ob\text{Id}), \sum \text{relationship}^{reTp}(re\text{Id}) \\ \sum \text{Role}_{reTp}^{roTp}(re\text{Id}, ro\text{Id}), \sum \text{Point}_{obTp}^{poTp}(ob\text{Id}, po\text{Id}) \\ \sum \text{Property}_{\text{nonPro}}^{\text{proTp}}(\text{nonproId}, \text{proId})\Big\}. \tag{3}$$

Definition 2: $\text{graph}_{Tp}(g\text{Id})$ refers to one model instance gId based on the metamodel of *Graph Tp*. In the model $\text{graph}_{Tp}(g\text{Id})$, $\text{object}^{obTp}(ob\text{Id})$ refers to the *Object* instance obId based on the metamodel of *Object obTp*. $\text{relationship}^{reTp}(re\text{Id})$ refers to the *Relationship* instance *re*Id based on the metamodel of *Relationship reTp*. $\text{Role}_{reTp}^{roTp}(re\text{Id}, ro\text{Id})$ refers to the *Role* instance *ro*Id based on the metamodel of *Role roTp* in the *Relationship*, *re*Id whose metamodel is *roTp*. $\text{Point}_{obTp}^{poTp}(ob\text{Id}, po\text{Id})$ refers to the *Point* instance *po*Id based on the meta-model of *Point poTp* in the individual *Object ob*Id, whose metamodel is *obTp*. $\text{Property}_{\text{nonPro}}^{\text{pro}Tp}(\text{nonProId}, \text{proId})$ refers to the property instance, proId, based on the metamodel of *Property* pro*Tp* in the *nonproperty* element, nonProId whose metamodel is one of *Graph*, *Object*, *Relationship*, *Role*, and *Point*. Through this definition, one model instance is constructed by its components and their links.

Definition 3: With the definition of *connector*, the concept *connection* is defined as a direct link between *Object* instances and *Point* instances in a *Graph* model instance, which is realized as one *Relationship* instance. Token $a =>b$ is defined as a *connection* that is linked from a to b, created based on two *connector* constraints. Thus, the connection *reTp*, refers to one link realized by the *Relationship* individual *reTp*, in the MBSE models, which is defined as follows:

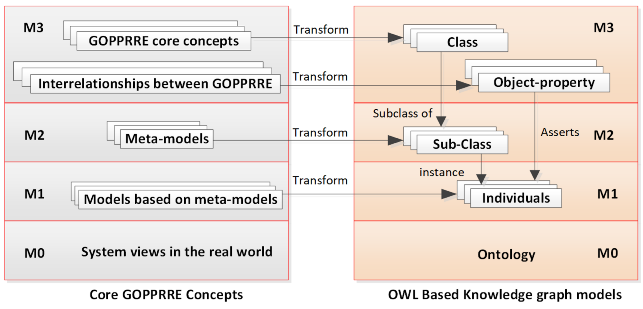


Fig. 5. Abstract syntax transformations.

$$\text{connection}_{reTp}(re\text{Id}) \\ = \{\text{connector}(\text{conId}') =>\text{connector}(\text{conId})\} \\ = \{\text{object}^{obTp'}(ob\text{Id}')(\vee \text{Point}_{obTp'}^{poTp'}(ob\text{Id},' po\text{Id}')) \\ \text{Role}_{reTp'}^{roTp'}(re\text{Id},' ro\text{Id}'), \text{relationship}^{reTp}(re\text{Id})\} \\ =>\{\text{relationship}^{reTp}(re\text{Id}) \\ \text{Role}_{reTp}^{roTp}(re\text{Id}, ro\text{Id}), \\ \text{object}^{obTp}(ob\text{Id})(\vee \text{Point}_{obTp}^{poTp}(ob\text{Id}, po\text{Id}))\} \tag{4}$$

where the connection is defined to represent a connecting through the *Relationship* instance *re*Id, whose *Relationship* type is $reTp$ from connector(conId′) to connector(conId). Through this definition, the directed links representing the logic in the MBSE models are demonstrated.

D. GOPPRRE Concept Mappings to Knowledge Graph Models

As shown in Fig. 5, a workflow for transforming the GOPPRRE core concepts to knowledge graph models based on OWL is demonstrated. The class for each GOPPRR concept represents the GOPPRRE meta–meta models (i.e., Graph, Object, Relationship, Role, Property, and Point). Their interrelationships are transformed to object property concepts in the knowledge graph model. Metamodels based on each GOPPRRE concept are then transformed to subclass concepts of the related GOPPRRE concept. Models are transformed to individuals based on their related subclasses. Based on the object property concepts, the interrelationships among individuals are defined. Moreover, the data property is used to define the value of each *Property*. The data property type is used to define the data type of each attribute. Finally, the MBSE models representing the real-world views are transformed to the ontology defined by the knowledge graph models using individuals, data properties, and object properties. The details of the transforming between MBSE models and OWL models are shown in Table II.

Apart from the abstract syntax, the concrete syntax of metamodels and models is described by the annotation and data properties as follows:

TABLE II
MBSE MODELS TRANSFORMING TO OWL MODELS

Concepts	MBSE models (SysML as example)	Ontology class in OWL models
Project and language for each model	One project for SysML	Project class
	One used language in a project, such as SysML	Language class
Meta-meta model	———	Graph class
		Object class
		Relationship class
		Role class
		Property class
		Point class
		Connector class
Meta-model	Requirement (Req) diagram	Subclass of Graph
	Requirement block for constructing a Req diagram	Subclass of Object
	Trace connection for constructing a Req diagram	Subclass of Relationship
	Two ends of the trace connection	Subclass of Role
	Attributes which can be defined in other meta-models, such as id	Subclass of Property
	Port used in an internal block diagram	Subclass of Point
Abstract syntax	A connecting rule collection for each graph	Subclass of Connector
Model	Instance of a Req diagram	Individual of the related subclass of Graph
	Instance of requirement block for constructing a Req diagram	Individual of the related subclass of Object
	Instance of trace connection for constructing a Req diagram	Individual of the related subclass of Relationship
	Instance of two ends of the trace connection	Individual of the related subclass of Role
	Instance of attributes which can be defined in other meta-models, such as id	Individual of the related subclass of Property
	Instance of port used in an internal block diagram	Individual of the related subclass of Point

1) *Annotation property:* Annotation property is used to represent the abstract syntax of metamodels, such as their original *icon paths*.
2) *Data property:* Data property is used to define the abstract syntax of models, such as the *icon path* of object instances in the models. This differs from the annotation property, because, when building MBSE models, the original icon of metamodels may be reconfigured.

IV. CASE STUDY

A case study is conducted to evaluate the designed MBSE ontology. Quantitative and qualitative approaches are separately applied [23], and the following two key measurements are considered.

1) The ontological completeness of the concrete syntax of MBSE formalisms.
   a) In the qualitative evaluation, SPARQL [33], a query language, is used to evaluate whether the ontology could completely represent the information generated from the MBSE models. Through the SPARQL query of knowledge graph models, which are generated from MBSE models, model components, and their relationships can be captured to identify whether all the components and relationships are stored in the knowledge graph models. To support this measurement, several metrics are defined, which are as follows:
      i) *Graph-include-objects (relationship)* refers to a situation, wherein one model includes all information related to its components or connections.
      i) *Object (relationship)-include-points (roles)* refers to a situation wherein one model component or connection includes all information related to its points or connection arrows.
      iii) *Object (graph and relationship)-include-properties* refers to a situation wherein one model (model component and connection) includes all information related to its attitudes.
   b) Using the quantitative approach, a DSM tool, *MetaGraph*, is developed to support the required ontology generation [34]. The number of key elements in the modeling languages and specifications, for which the ontology is formalized, are analyzed to evaluate its completeness.
2) Ontology logic related to the abstract syntax of MBSE formalisms.
   a) In the qualitative evaluation, SQWRL [35], a query language, is used to evaluate its description logic, by querying OWL as its semantic web-rule language to design the rules needed to assign the *subject*, *predicate*, and *object* by their defined *predicates* [36]. It is adopted to evaluate whether the ontology could capture the information needed to define the abstract syntax of the MSBE models. Through the SQWRL query, the relationships and their two ends are identified to verify whether all the relationships are connected correctly. The following two metrics are thus considered.

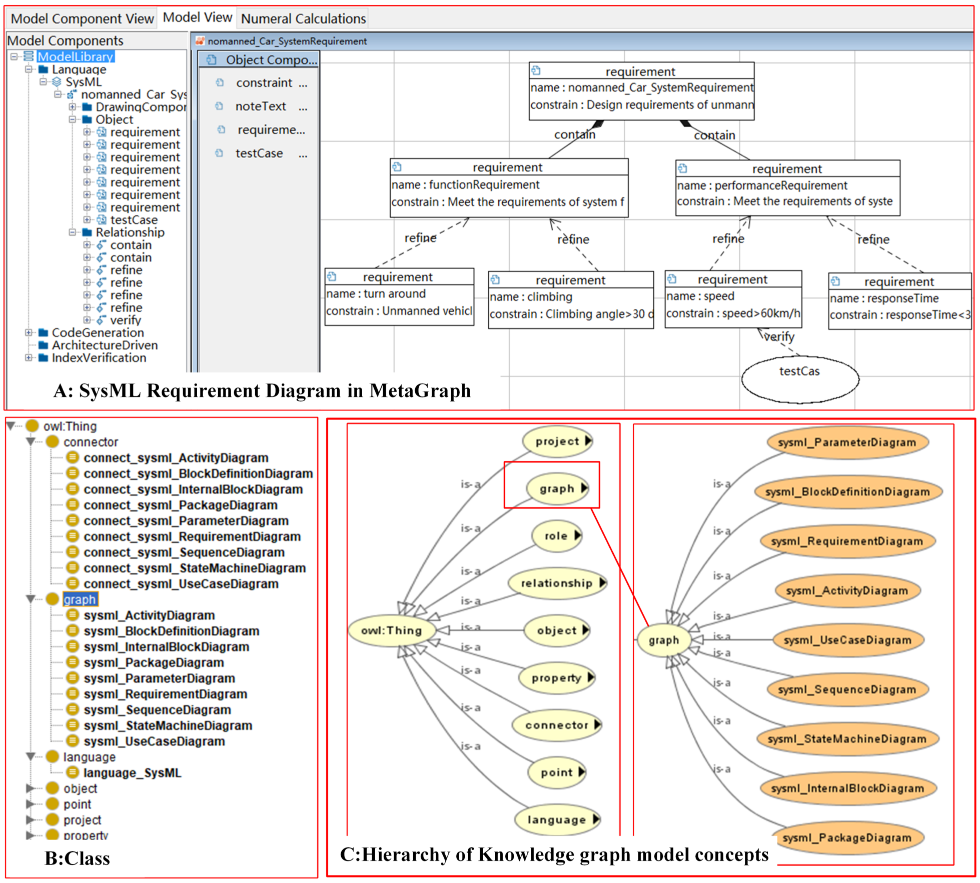

A: SysML Requirement Diagram in MetaGraph
B:Class
C:Hierarchy of Knowledge graph model concepts
owl:Thing
graph
project
role
relationship
object
property
connector
point
language
is-a

Fig. 6. Knowledge graph modeling using *MetaGraph*. (A) One SysML model in *MetaGraph*. (B) Knowledge graph model class defined based on the proposed formalisms in Protégé. (C) Ontology class hierarchy in the knowledge graph models.

i) *Relationship definitions in the MBSE model:* They present the connections (logic flows) with two ends in the MBSE models. The connections between model components or points define the basic logic for constructing one MBSE model.

ii) *Direction of relationship:* This presents the start of each connection, which decides how the connection is linked to its two sides.

iii) *Connecting patterns:* The connecting patterns mentioned in the next section are evaluated to verify if the ontology can represent all the related models across the MBSE languages.

b) Using the quantitative approach, *MetaGraph* was used to generate knowledge graph models, wherein the number of graphs could support connection rules of different modeling languages and specifications. They were identified to evaluate the logic supported by the designed ontology.

Quantitative and qualitative analyses were performed to evaluate the completeness of the concrete syntax and logic of the abstract syntax. During quantitative analysis, a DSM tool, *MetaGraph*, was developed to evaluate the ontology using several MBSE languages [34], as shown in Fig. 6. Several metamodels were developed with the *MetaGraph* based on five existing MBSE language specifications. In the qualitative approach, SQWRL and SPARQL were used to evaluate the completeness and logic of the developed ontology through reasoning. Currently, we mainly focus on the reasoning based on SQWRL and SPARQL queries, whereas OWL reasoning is not considered in this article.

A. Quantitative Analysis

When implementing the quantitative analysis, the *MetaGraph* was used to develop MBSE models based on the proposed ontology [37]. As shown in Fig. 6(A), a requirement diagram model is demonstrated. Within *MetaGraph*, *Object* and *Relationship* metamodels of the SysML requirement diagram are used to construct this model. As shown in Fig. 6(B), ontology classes of the entire SysML metamodels are shown. The concepts of *Language*, *Graph*, *Object*, *Relationship*, *Point*, *Role*, and *Property* are generated from the metamodels of SysML specifications through the *MetaGraph* plugin. As shown in Fig. 6(C), the hierarchy of the knowledge graph model is demonstrated. Based on the MBSE formalisms, the knowledge graph models are generated from metamodels and models in *MetaGraph*.

Five MBSE languages were developed to evaluate whether the ontology could provide enough information for the MBSE model constructions.[4] As listed in Table III, metamodels of five general MBSE languages were built to compare four existing

[4]See the ontology models in *MetaGraph:* https://www.youtube.com/watch?v=o2SZKMYn5PQ. The ontology model and MBSE model

TABLE III
METAMODELS IN FIVE EXISTING MBSE LANGUAGES DESCRIBED BY THE GOPPRRE APPROACH

Language	Graph	Object	Point	Property	Relationship	Role	Referred tools
SysML	9(9)	73(64)	11(11)	10(10)	31(31)	31(31)	(Magic draw) [38]
BPMN	1(1)	81(77)	0(0)	51(46)	5(3)	10(0)	(BPM Camunda) [39]
UPDM (Unified Profile for DoDAF/MODAF)	52(52)	100 (123)	7(6)	84 (96)	57 (50)	54 (0)	(Magic draw) [38]
EASTADL	10(9)	67(62)	17(17)	93(93)	23(21)	68(64)	(MetaEdit+) [10]
OPM	1(1)	3(3)	0(0)	9(8)	15(15)	12(4)	(OPCAT) [40]

TABLE IV
CONNECTION RULES FOR THE EXISTING LANGUAGES COMPARED WITH CONNECTORS IN THE GOPPRRE APPROACH

Language	Connection rules	Connectors
SysML	31	62
BPMN	72	133
UPDM	396	792
EASTADL	215	233
OPM	39	54

tools. From the results, the ontology could formalize almost all the metamodels of the related languages. All *Graph* metamodels were developed based on the five MBSE language specifications. Some objects were different from the existing tools, because some elements in their tools were not defined as objects in our approach. For example, in Magic draw, some properties were defined as elements in their diagram-building environment so that the users could easily configure an object's property. The concrete syntax of all the languages were completely transformed to the developed ontology in *MetaGraph*.

Apart from the concrete syntax, the abstract syntax was also evaluated by comparing the connection rules with different languages in other tools. In the *MetaGraph*, the connectors between relationships and objects were compared with the rules for connecting different elements in other tools. This was done to determine whether the ontology can formalize the logic flow between different MBSE model elements. As listed in Table IV, connection rules refer to the specifications used to define how to connect model compositions or their ports in the five existing languages by other tools. The *connectors* were used to create connections between *Objects* or their *Points* in our approach. From the results, almost all connection rules were defined based on the given ontology, although the number of *connectors* were not all twice the connection rules in the MBSE languages. This occurred because of the discrepancies of constructing the *Graph* metamodels. For example, in the BPMN, the number of connectors was 11 fewer than twice the number of connections, because the links between the text *Object* and other 12 *Objects* in the BPMN specification required one *Role* for the text *Object* and 12 roles for the other *Objects* in our approach, compared with the 12 connection rules in other tools.

B. Qualitative Analysis

To qualitatively verify the ontology, SPARQL and SQWRL were used to evaluate the completeness and logic of the MBSE models through reasoning. To test the generated knowledge graph model [see Fig. 6(B) and (C)] from the SysML requirement diagram model in Fig. 6(A), the completeness and logic in the knowledge graph model were evaluated separately using Algorithms 1 and 2.

Algorithm 1: SPARQL Algorithm for Verifying the Completeness of the MBSE Models.

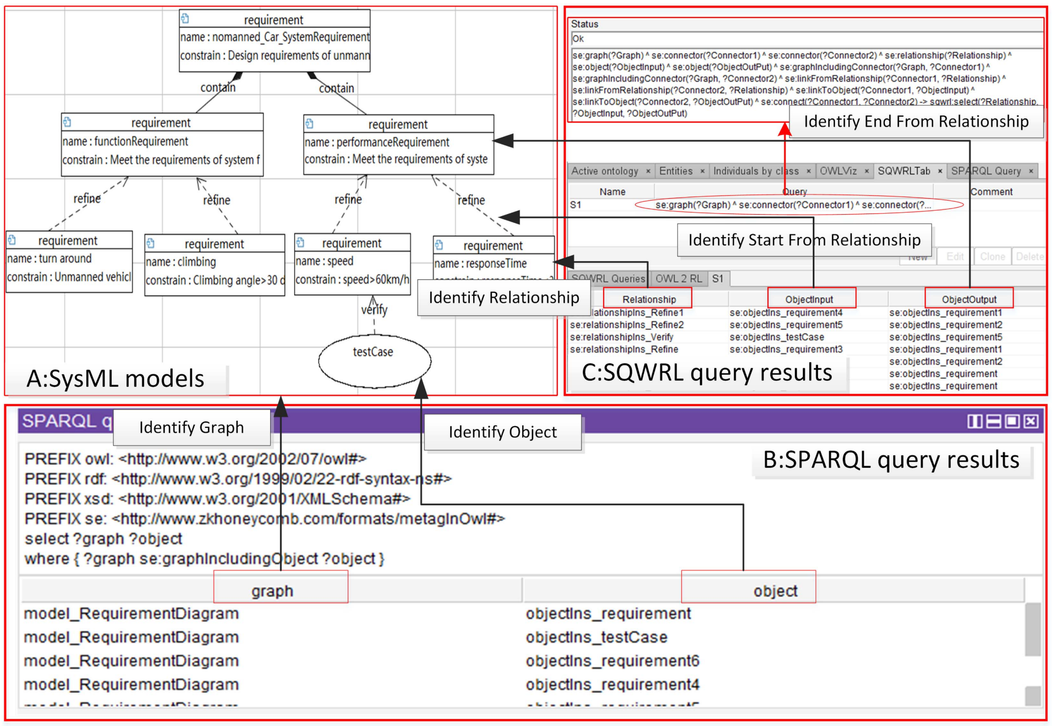
```
PREFIXowl :< http : //www.w3.org/2002/07/owl# >
PREFIXrdf :< http : //www.w3.org/1999/02/22 − rdf
  −syntax − ns# >
PREFIXxsd :< http : //www.w3.org/2001/
  XMLSchema# >
PREFIXse < http : //www.zkhoneycomb.com/formats/
  metagInOwl# >

// If Graph includes Objects(Relationships)
select ?graph ?object ?relationship
where {
?graph se:graphIncludingObject(graphIncludingRelationship)
  ?object(relationship)
}

// If Objects(Relationships) includes Points(Roles)
select ?object ?point ?relationship ?role
where {
?object(relationship) se:linkObjectAndPoint(linkRelationship
  AndRole) ?point(role)
}
// If Object(Graph and Relationship) includes Properties
select ?graph ?object ?point ?relationship ?role ?property
where {
?graph(object, point, relationship or role) se:hasProperty
  ?Property
}
```

Algorithm 1 is a SPARQL query algorithm developed to verify the completeness of the generated ontology. As shown in Fig. 6, the knowledge graph model generated from the SysML model was used to verify the completeness of the ontology. Based on the Algorithm 1, the SPARQL script was developed to verify the three metrics mentioned in Section II. As shown in Fig. 7(B), the query results demonstrated that all *Object* and *Relationship* instances representing the SysML model were captured to describe its model structure. Moreover, *Property* instances were also identified in different metamodels. From the results, we can infer that the completeness of the ontology was verified because all the information related to the SysML model was completely transformed into the knowledge graph model.

for the case study are in gitee:https://gitee.com/zkhoneycomb/open-share/blob/master/Papers/Design%20Ontology%20Supporting%20Model-based%20Systems%20Engineering%20Formalisms/.keep



```
Status
Ok
se:graph(?Graph) ^ se:connector(?Connector1) ^ se:connector(?Connector2) ^ se:relationship(?Relationship) ^
se:object(?ObjectInput) ^ se:object(?ObjectOutPut) ^ se:graphIncludingConnector(?Graph, ?Connector1) ^
se:graphIncludingConnector(?Graph, ?Connector2) ^ se:linkFromRelationship(?Connector1, ?Relationship) ^
se:linkFromRelationship(?Connector2, ?Relationship) ^ se:linkToObject(?Connector1, ?ObjectInput) ^
se:linkToObject(?Connector2, ?ObjectOutPut) ^ se:connect(?Connector1, ?Connector2) -> sqwrl:select(?Relationship,
?ObjectInput, ?ObjectOutPut)
```

Name	Query	Comment
S1	se:graph(?Graph) ^ se:connector(?Connector1) ^ se:connector(?...	

Relationship	ObjectInput	ObjectOutput
se:relationshipIns_Refine1	se:objectIns_requirement4	se:objectIns_requirement1
se:relationshipIns_Refine2	se:objectIns_requirement5	se:objectIns_requirement2
se:relationshipIns_Verify	se:objectIns_testCase	se:objectIns_requirement5
se:relationshipIns_Refine	se:objectIns_requirement3	se:objectIns_requirement1
		se:objectIns_requirement2
		se:objectIns_requirement
		se:objectIns_requirement

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX se: <http://www.zkhoneycomb.com/formats/metagInOwl#>
select ?graph ?object
where { ?graph se:graphIncludingObject ?object }
```

graph	object
model_RequirementDiagram	objectIns_requirement
model_RequirementDiagram	objectIns_testCase
model_RequirementDiagram	objectIns_requirement6
model_RequirementDiagram	objectIns_requirement4

Fig. 7. SPARQL and SQWRL query results.

Algorithm 2: SQWRL Algorithm for Verifying the Logic of the MBSE Models.

```
// Query relationships in the MBSE models
graph(?Graph) Λ connector(?Connector1) Λ
 connector(?Connector2) Λ relationship(?Relationship)
Λ object(?ObjectInput) Λ object(?ObjectOutput)
Λ graphIncludingConnector(?Graph, ?Connector1)
Λ graphIncludingConnector(?Graph, ?Connector2)
Λ linkFromRelationship(?Connector1,?Relationship)
Λ linkFromRelationship(?Connector2,?Relationship)
Λ linkToObject(?Connector1,? ObjectInput) Λ
linkToObject(?Connector2, ?ObjectOutPut) Λ
connect(?Connector1, ?Connector2)
− > sqwrl:select(?Relationship,?ObjectInput,
 ?ObjectOutput)
// Query the direction of each relationship
− > sqwrl:select(?Graph, ?Relationship, ?ObjectInput)
```

Algorithm 2 is a SQWRL algorithm used to verify the logic flows in the given SysML model. To capture the *connections* among object$^{obTp}(ob\text{Id})$, relationship$^{reTp}(ob\text{Id})$, and point$^{reTp}(ob\text{Id}, po\text{Id})$, which are defined as the individuals representing the *Object*, *Relationship*, and *Point* concepts in the model, Algorithm 2 was used to capture the related information. All individuals representing the SysML model were queried using the object properties listed as follows.

1) *graphIncludingConnector* refers to the connector developed in a graph associated with *Relationship*, *Role*, and *Object* (*Points*), as shown in (2).
2) *linkFromRelationship* refers to the relationship linked to one connector, where one *Relationship* has one *Role* as its end, as described by *linkRelationshipAndRole*.
3) *linkToObject* refers to the connector linked to one *Object*, where one *Role* is connected to one *Object* or one *Point* described by *roleBindingObject* or *roleBindingPoint*. If *Points* are not involved in the connection, the *Object* is defined as the end of the relationship or vice versa.
4) *connect* refers to one *connector* (start) linked to another *connector* (end). It is used to describe the direction of the relationship.

As shown in Fig. 7(C), the query results identify the *Relationship* individuals between different *Object* individuals and *Point* individuals. Moreover, the direction of the *Relationship* is identified based on its starting role from *Object* individuals.

In summary, the connecting patterns are constructed based on the proposed ontology, which are used to generate the knowledge graph models. Through queries of these models by Algorithms 1 and 2, the MBSE models based on such connecting patterns can be constructed by the proposed ontology. Through Algorithm 1, the SPARQL query results identify the mappings between *Graph* individuals and *Object* individuals. All related *Object* and *Relationship* instances in the SysML model are captured. Using Algorithm 2, all sets of the *Relationship* individuals with their Start *Role* and End *Role* individuals are identified. Through these sets, we can understand how the *Relationship* individuals are connected in the SysML model. Finally, all the connecting patterns are supported by the proposed ontology as shown in Table V. From the table, we find this ontology can support different modeling scenarios under the given connecting patterns across the MBSE languages.

TABLE V
CONNECTING PATTERN EVALUATION

Connecting patterns	Evaluation	Scenario demonstration
Reference relationship	Yes	Relationship without properties
Binary relationship	Yes	Relationship with only two connectors for two separated objects
N-ary relationship	Yes	Such as SysML requirement diagram
Set relationship	Yes	The same relationships in different diagrams
Role relationship	Yes	Such as class diagram
Role-point relationship	Yes	Such as SysML internal block diagram

C. Discussion

When evaluating the proposed ontology, internal validity and external validity are considered. Internal validity refers to establishing a cause-and-effect cue between the given ontology and the proposed MBSE formalisms. From the quantitative and qualitative analyses, through *MetaGraph*, five existing MBSE languages are proposed to implement the internal validity. The results proved that the ontology based on the GOPPRRE approach could formalize at least five MBSE modeling languages, which are commonly used to model systems of systems (e.g., UPDM), system architectures (e.g., SysML), business processes (e.g., BPMN), and domain-specific knowledge for the architectural description of automotive embedded systems. External validity refers to how well the ontology can be used for other modeling languages.

In this case study, the generated knowledge graph models captured all the information of the abstract syntax and concrete syntax of the MBSE metamodels and models. The visual representations of the metamodels and models are covered in the knowledge graph models, such as icon path and shape. Through such information, *MetaGraph* can represent the MBSE metamodels and models by their own visual representations. Moreover, the connections between different MBSE model components are included in the knowledge graph models. With this information, *MetaGraph* can represent the direct links between different model components in the MBSE models.

In addition to the MBSE formalisms, this ontology also empowers data interoperability. *GOPPRR* is a powerful approach to describe domain-specific characteristics, whose meta–meta models have excellent descriptive capabilities [31]. The results listed in Tables III and IV prove that the current GOPPRRE ontology could integrate at least five existing MBSE languages. It can be used as the middleware for the MBSE community to support data exchange among these languages. Furthermore, because of the unified ontology based on the same meta–meta models, when developing the model instances across different graphs, such instances can be described in a unified knowledge graph model. The given ontology provides a specification for knowledge graph models across modeling tools to improve their interoperability. Moreover, the integrated ontology across knowledge graph models can mitigate the risks of tool integration and model integration failures when new modeling tools are used for system development.

The traditional model-driven approaches used to support model transformation through code generation [29]. The proposed ontology enables to provide one specification across modeling tools without code generation. Using this ontology, all the heterogeneous models from different tools can be integrated in one unified knowledge graph modeling platform. Furthermore, the OWL models, generated from the MBSE models, are widely used to support ML and AI techniques. It supports reasoning and analyzing of the target modeling systems. For example, Algorithms 1 and 2 can be used to capture information from the MBSE models for knowledge management. The knowledge graph models generated from the MBSE models can be directly used to construct cognitive digital twins to support decision making during the system development and operations [27]. Such models are considered as middleware for capturing required data through reasoning when training and verifying AI and ML algorithms for decision makings. The models provide more potential capabilities for obtaining related data through queries.

Compared with the existing outcomes presented in [37], several promotions have been conducted in this article, which are as follows.

1) Based on the proposed GOPPRR concepts, we provide a unified formalism for MBSE models.
2) A new concept “extension” is provided to identify the constraints when constructing the meta-models of *Graph*.
3) Modeling languages, such as SysML and BPMN, are evaluated in this article.

There are also several limitations in this article, which need more efforts in future research.

1) More MBSE languages need to be evaluated in future studies.
2) The MBSE formalisms in this article do not include the decomposition and explosion sections, which will be extended for describing the hierarchy of MBSE models.
3) The coherence and interoperability across graphs are not considered in this article, thus, more research on extensions of GOPPRRE will be done for describing the interrelationships across different graph models.

Scalability is a critical issue regarding the application of the developed ontology. Thus, the ontology has been adopted by the Industrial Ontologies Foundry Systems Engineering Working Group to be discussed (https://www.industrialontologies.org/). Moreover, the proposed ontology will be integrated with Basic Formal Ontology as a systems engineering ontology extension [41], [42] to improve the scalability.

V. CONCLUSION

In this article, we designed an ontology based on the GOPPRRE approach that supports MBSE formalisms using OWL for model integration. First, we demonstrated the GOPPRRE concepts based on an M0–M3 modeling framework. Then, we developed a transformation rule between GOPPRRE concepts and ontology concepts in OWL models. Based on the transformation rules, OWL models were generated from five existing MBSE languages by a DSM tool *MetaGraph*. Qualitative and quantitative approaches were used to evaluate the completeness and logic of the generated knowledge graph models. The results proved that the designed ontology can support MBSE formalisms, showing the potential of this method to become the standardized approach for the MBSE community in the future.

ACKNOWLEDGMENT

The authors would like to thank Mr. H. Wang for his contributions during his master thesis.

REFERENCES

[1] M. Kharrat, "Integration of electromagnetic constraints as of the conceptual design through an MBSE approach," *IEEE Syst. J.*, vol. 15, no. 1, pp. 747–758, Mar. 2021.

[2] F. A. Halstenberg and R. Stark, "Study on the feasibility of modelling notations for integrated product-service systems engineering," *Procedia CIRP*, vol. 83, pp. 157–162, 2019.

[3] T. McDermott, D. DeLaurentis, P. Beling, M. Blackburn, and M. Bone, "AI4SE and SE4AI: A research roadmap," *Insight*, vol. 23, no. 1, pp. 8–14, 2020.

[4] J. Hao, L. Xu, G. Wang, Y. Jin, and Y. Yan, "A knowledge-based method for rapid design concept evaluation," *IEEE Access*, vol. 7, pp. 116835–116847, 2019.

[5] C. E. Dickerson and D. Mavris, "A brief history of models and model based systems engineering and the case for relational orientation," *IEEE Syst. J.*, vol. 7, no. 4, pp. 581–592, Dec. 2013.

[6] L. Bassi, C. Secchi, M. Bonfe, and C. Fantuzzi, "A SysML-based methodology for manufacturing machinery modeling and design," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 6, pp. 1049–1062, Dec. 2011.

[7] Y. Mordecai, O. Orhof, and D. Dori, "Model-based interoperability engineering in systems-of-systems and civil aviation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 4, pp. 637–648, Apr. 2018.

[8] G. Ravikumar, S. A. Khaparde, and R. K. Joshi, "Integration of process model and CIM to represent events and chronology in power system processes," *IEEE Syst. J.*, vol. 12, no. 1, pp. 149–160, Mar. 2018.

[9] J. Bezivin and O. Gerbe, "Towards a precise definition of the OMG/MDA framework," in *Proc. 16th Annu. Int. Conf. Automated Softw. Eng.*, 2005, pp. 273–280. [Online]. Available: http://ieeexplore.ieee.org/document/989813/

[10] S. Kelly, K. Lyytinen, and M. Rossi, *Advanced Information Systems Engineering* (Lecture Notes in Computer Science), vol. 1080, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, Eds. Berlin, Germany: Springer, 1996.

[11] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proc. IEEE*, vol. 103, no. 11, pp. 2104–2132, Nov. 2015.

[12] G. Ćwikła, A. Gwiazda, W. Banaś, Z. Monica, and K. Foit, "Analysis of the possibility of SysML and BPMN application in formal data acquisition system description," *IOP Conf. Series Mater. Sci. Eng.*, vol. 227, Aug. 2017, Art. no. 012034.

[13] C.-J. Sjöstedt *et al.*, "Developing dependable automotive embedded systems using the EAST-ADL; representing continuous time systems in SysML," in *Proc. 1st Int. Workshop Equ.-Based Object-Oriented Lang. Tools*, 2007, pp. 25–36.

[14] M. E. Morales-Trujillo, B. Escalante-Ramírez, P. Ángeles, H. Oktaba, and G. Ibargöengoitia-González, "Towards a representation of enterprise architecture based on Zachman framework through OMG standards," in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, 2018, pp. 225–224.

[15] C. J. Paredis *et al.*, "5.5.1 an overview of the SysML-Modelica transformation specification," *Proc. INCOSE Int. Symp.*, vol. 20, no. 1, pp. 709–722, Jul. 2010.

[16] N. Papakonstantinou and S. Sierla, "Generating an object oriented IEC 61131-3 software product line architecture from SysML," in *Proc. IEEE 18th Conf. Emerg. Technol. Factory Automat.*, Sep. 2013, pp. 1–8.

[17] W. Wang, N. Niu, M. Alenazi, and L. Da Xu, "In-place traceability for automated production systems: A survey of PLC and SysML tools," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3155–3162, Jun. 2019.

[18] T. Walter, F. S. Parreiras, and S. Staab, "An ontology-based framework for domain-specific modeling," *Softw. Syst. Model.*, vol. 13, no. 1, pp. 83–108, Feb. 2014.

[19] Z. Ming *et al.*, "Ontology-based representation of design decision hierarchies," *J. Comput. Inf. Sci. Eng.*, vol. 18, no. 1, Mar. 2018, Art. no. 011001.

[20] B. Beihoff *et al.*, "A world in motion systems engineering vision 2025," *Proc. Int. Council Syst. Eng.*, vol. 327, no. 5970, pp. 1183–1183, Mar. 2010.

[21] L. Yang, K. Cormican, and M. Yu, "Ontology learning for systems engineering body of knowledge," *IEEE Trans. Ind. Inform.*, vol. 17, no. 2, pp. 1039–1047, Feb. 2021.

[22] C. Seidner and O. Roux, "Formal methods for systems engineering behavior models," *IEEE Trans. Ind. Inform.*, vol. 4, no. 4, pp. 280–291, Nov. 2008.

[23] J. Lu, G. Wang, and M. Torngren, "Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain," *IEEE Syst. J.*, vol. 14, no. 1, pp. 1297–1308, Mar. 2020.

[24] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, "Experimentable digital twins-streamlining simulation-based systems engineering for industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 14, no. 4, pp. 1722–1731, Apr. 2018.

[25] R. Wang, A. B. Nellippallil, G. Wang, Y. Yan, J. K. Allen, and F. Mistree, "Ontology-based uncertainty management approach in designing of robust decision workflows," *J. Eng. Des.*, vol. 30, no. 10–12, pp. 726–757, Dec. 2019. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09544828.2019.1668918

[26] J. Hao, Y. Yan, L. Gong, G. Wang, and J. Lin, "Knowledge map-based method for domain knowledge browsing," *Decis. Support Syst.*, vol. 61, pp. 106–114, May 2014.

[27] J. Lu, X. Zheng, A. Gharaei, K. Kalaboukas, and D. Kiritsis, "Cognitive twins for supporting decision-makings of internet of things systems," in *Proc. 5th Int. Conf. Ind. 4.0 Model Adv. Manuf.*, L. Wang, V. D. Majstorovic, D. Mourtzis, E. Carpanzano, G. Moroni, and L. M. Galantucci, Eds. Cham, Switzerland: Springer, 2020, pp. 105–115.

[28] S. Guo, J. Wang, Z. Chen, Z. Lu, J. Guo, and L. Yang, "Security-aware task mapping reducing thermal side channel leakage in CMPs," *IEEE Trans. Ind. Inform.*, vol. 15, no. 10, pp. 5435–5443, Oct. 2019.

[29] J. Lu, D. Gürdür, D.-J. Chen, J. Wang, and M. Törngren, "Empirical-evolution of frameworks supporting co-simulation tool-chain development," in *Proc. Adv. Intell. Syst. Comput.*, 2018, vol. 745, pp. 813–828.

[30] J. Chen, Y. Ruan, L. Guo, and H. Lu, "BCVehis: A blockchain-based service prototype of vehicle history tracking for used-car trades in China," *IEEE Access*, vol. 8, pp. 214 842–214851, 2020.

[31] H. Kern, A. Hummel, and S. Kühne, "Towards a comparative analysis of meta-meta models," in *Proc. Compilation Co-Located Workshops DSM11*, 2011, pp. 7–12.

[32] S. V. Mierlo, Y. V. Tendeloo, B. Meyers, and H. Vangheluwe, *Domain-Specific Modelling for Human-Computer Interaction,* (Human-Computer Interaction Series), B. Weyers, J. Bowen, A. Dix, and P. Palanque, Eds., Cham, Switzerland: Springer, 2017.

[33] E. Patti *et al.*, "Event-driven user-centric middleware for energy-efficient buildings and public spaces," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1137–1146, Sep. 2016.

[34] L. Jinzhi, W. Guoxin, M. Junda, K. Dimitris, Z. Hang, and T. Martin, "General modeling language to support model-based systems engineering formalisms (Part 1)," in *Proc. INCOSE Int. Symp.*, 2020, pp. 323–338.

[35] M. O'Connor and A. Das, "SQWRL: A query language for OWL," in *Proc. CEUR Workshop*, 2009, pp. 208–215.

[36] I. Horrocks *et al.*, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member Submission*, vol. 21, no. 79, pp. 1–31, 2004.

[37] H. Wang, G. Wang, J. Lu, and C. Ma, "Ontology supporting model-based systems engineering based on a GOPPRR approach," in *Proc. 7th World Conf. Inf. Syst. Technol.*, Cham, Switzerland: Springer, 2019, pp. 426–436.

[38] M. Chami, A. Aleksandraviciene, A. Morkevicius, and J.-M. Bruel, "Towards solving MBSE adoption challenges: The D3 MBSE adoption toolbox," in *Proc. INCOSE Int. Symp.*, vol. 28, no. 1, pp. 1463–1477, Jul. 2018.

[39] A. Fernandez, "Camunda BPM platform loan assessment process lab," Queensland Univ. Technol., Brisbane, QLD, Australia, Tech. Rep. 8725853, 2014.

[40] D. Dori, *Model-Based Systems Engineering With OPM and SysML*. New York, NY, USA: Springer, 2016. [Online]. Available: http://link.springer.com/10.1007/978-1-4939-3295-5

[41] R. Arp, S. Barry, and A. D. Spear, *Building Ontologies With Basic Formal Ontology*. Cambridge, MA, USA: MIT Press, 2015.

[42] B. S. Kulvatunyou, E. Wallace, D. Kiritsis, B. Smith, and C. Will, "The industrial ontologies foundry proof-of-concept project," in *Proc. Adv. Prod. Manage. Syst. Smart Manuf. Ind. 4.0*, I. Moon, G. M. J. Lee D. Park Kiritsis, and G. von Cieminski, Eds. Cham, Switzerland, 2018, pp. 402–409.