Journal of Systems Engineering and Electronics
Vol. 31, No. 3, June 2020, pp.527 – 538

An ontological metamodeling framework for semantic simulation model engineering

LEI Yonglin, ZHU Zhi*, and LI Qun

School of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Abstract: Recently, the ontological metamodel plays an increasingly important role to specify systems in two forms: ontology and metamodel. Ontology is a descriptive model representing reality by a set of concepts, their interrelations, and constraints. On the other hand, metamodel is a more classical, but more powerful model in which concepts and relationships are represented in a prescriptive way. This study firstly clarifies the difference between the two approaches, then explains their advantages and limitations, and attempts to explore a general ontological metamodeling framework by integrating each characteristic, in order to implement semantic simulation model engineering. As a proof of concept, this paper takes the combat effectiveness simulation systems as a motivating case, uses the proposed framework to define a set of ontological composable modeling frameworks, and presents an underwater targets search scenario for running simulations and analyzing results. Finally, this paper expects that this framework will be generally used in other fields.

Keywords: ontology, metamodeling, semantic composability, model-driven engineering (MDE).

DOI: 10.23919/JSEE.2020.000032

1. Introduction

In the recent past, ontology as a means of describing a diversified simulation system has taken an increasingly significant role at the heart of the conceptual modeling phase in system engineering. It allows the users to describe the reality in an analytical way. Hence, many researchers in modeling and simulation (M&S) have paid much attention to it, especially in the area of model-driven engineering (MDE). These researchers usually focus on employing it to capture domain knowledge, and the adapted know-how is considered as a key attempt to gain efficiency and productivity as well as to improve the product quality [1]. Meanwhile, both ontology and model are shared by a group of people, and are usually represented in different forms due to different preferences of simulation modelers. Consequently, these differences raise many issues and make the model composability [2], particularly the semantic composability, very difficult.

According to [3], model composability has four different levels in general: deep semantic composability (also called full composability), semantic composability, syntactical composability, and no composability. Regarding different types of model heterogeneity and different ways of semantic mapping and matching, syntactical and semantic composability can be divided into further categories [4]. Traditional technological standard specifications such as the high level architecture (HLA) [5], simulation model portability (SMP) [6] and discrete event system specification (DEVS) [7,8] mainly focus on the syntactical facet of model composability. These specifications aim to build a commonly accepted standard to represent models. However, without any consideration on the semantic facet, models are hard to be integrated meaningfully. Therefore, some domain specific simulation systems or platforms such as extended air defense system (EADSIM) [9], system effectiveness analysis simulation (SEAS) [10], and joint mission effectiveness analysis simulator for utility, research and evaluation (JointMEASURE) [11] concentrate on domain knowledge abstraction from the domain knowledge's perspective, and get considerable successful applications. All these attempts lay a well foundation for realizing semantic composability of simulation models and also provide a lot of experience for complex systems M&S. Yet, given a certain domain, a set of composable modeling frameworks (CMF) is the key to engineering semantic composability of simulation models [12].

This paper applies ontology in CMF, also called ontology-aware CMF (ontoCMF), to enhance CMF's semantic expressiveness. The remainder is structured as follows. Section 2 describes the background. In Section 3, the general ontological metamodeling framework is proposed.

Manuscript received December 18, 2019.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (61273198).

Section 4 shows how to build the ontoCMF. A typical case study of combat effectiveness simulation systems (CESS) is illustrated in Section 5, which is followed by conclusions in Section 6.

2. Background

2.1 Model, ontology and metamodel

This study adopts a definition in [13] that a model is a representation of reality for some definite purpose. A model can be used to represent many different kinds of realities, such as domains and systems. According to [14], a model of a system is a description or specification of that system and its environment for some certain purpose, where the environment of a system is described by a domain model. Hence, a model has two distinctive categories: descriptiveness and prescriptiveness [15]. A descriptive model describes reality, but the reality is not constructed from it, that is, the model truth lies in reality. Whereas, a prescriptive model prescribes the structure or behavior of the reality and the reality is constructed according to the model, so in a descriptive model the truth lies in the model itself [16].

According to [17], ontology is a shared, descriptive, structural model, representing reality by a set of concepts, their interrelations, and constraints under the open-world assumption (OWA). In the OWA, things that have not been represented explicitly are assumed unknown, whereas the close-world assumption (CWA) is important to restrict arbitrary extensions which could incur consistencies, underlying that what has not been specified is either implicitly disallowed or implicitly allowed. In general, there are two types of ontology: reference ontology and local ontology. In the former case, the ontology is more stable and is usually released by standard organizations, to prescribe how the ontologies on the next level below, i.e., local ontology, are represented. In the latter case, the ontology extends/specializes certain concepts or relationships of the reference ontology on an upper level, to achieve knowledge reuse among different applications. In practice, ontologies usually include top ontology, domain ontology, business ontology and application ontology. Firstly, the top ontology describes the most common concepts and relationships, like objects, events, and time, etc., which are independent of concrete applications. Secondly, the domain ontology and the business ontology describe the system structure and behavioral logic for a given domain. The former emphasizes the explicit representation of the static domain knowledge, whereas the latter concentrates on concepts and relationships for a certain task or behavior. Thirdly, the application ontology is often customized as a concrete scenario.

A metamodel is a representation for a class of models [18,19]. Models that conform to the metamodel are true so the metamodel is a prescriptive model. The term “meta” is widely used combined with concepts such as computing, language, and data, forming new words like metacomputing, metalanguage, metadata and so on. These terms represent some implications like “super”, “fundamental” and “specification” from an abstract perspective. In general, it is reasonable to say that a metamodel is the model that prescribes models, metadata is the data that schedule data, and metalanguage is the language that specifies languages. In metamodeling, it is helpful to distinguish similarity relations for precisely locating a special concept within the similarity space. This study agrees with the classification of similarity relations in [17] and makes a simple extension for the instanceOf relation, distinguishing ontological instanceOf and linguistic instanceOf relations, as shown in Fig. 1.

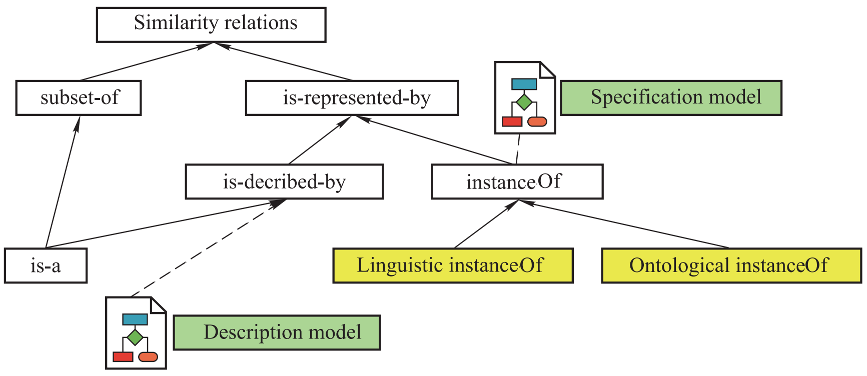

Fig. 1 Similarity relations with linguistic and ontological extensions

2.2 Ontological metamodel

The ontological metamodel integrates ontology and metamodel, both of which are closely related but also independent of each other. On the one hand, the ontological metamodel is based on the description logic so it has formalized features to some degree. Moreover, it directly uses domain concepts and relationships to describe systems so it is closer to human's way of thinking; thus it is easier to be understood. On the other hand, the ontological metamodel is a kind of meta-theory that improves the level of system abstractions to prescribe the representation of multiple system objects. As a result, both ontology and metamodel lay the foundational theory of ontological metamodeling, enable the semantic expressiveness of metamodels and hence promote semantic composability of simulation models [20]. In general, the ontology is introduced through two ways: ontological metamodeling from scratch and unified modeling language (UML) extension with tags.

(i) Ontological metamodeling from scratch is also called the implantation way. This way directly introduces a descriptive ontology concept as the metaclass of a certain metamodel. For example, one builds an ontological metamodel by using the web ontology language (OWL) to describe the concepts and structure of metamodels. In this way, the consequent ontological metamodel is usually clumsy because the metamodel and ontology are strongly coupled, and incurring a little evolution of the ontology may change the overall ontological metamodel structure. Therefore, it belongs to a heavyweight way and is more suitable for the case of a considerable stable ontology, such as the top ontology or the reference ontology.

(ii) UML extension with tags is also called the tag way. This way defines a set of properties for the metaclasses of a given metamodel by several descriptive ontology concepts. Using the UML class diagram to construct a metamodel, for example, one tags the properties of a metaclass by defining its connection to the ontology concept. In this way, the metamodel and ontology are loosely coupled so that a certain ontology evolution will not change the metamodel. Hence, it is a lightweight way to be usually used to build the local ontology, such as the domain ontology, the business ontology and the application ontology.

2.3 Ontological metamodeling for semantic composability

As discussed before, traditional methods for model composability focus on the syntactical facet from a technological point of view. Consequently, many simulation modelers concentrate on the simulation software, hardware and also the network environment, but pay less attention to how the information contained in simulation models are represented and understood, neither is it paid to composing simulation models effectively and meaningfully for a special simulation service. From a domain's perspective, ontological metamodeling that is driven by realities or the domain knowledge focuses on the semantic facet. As such, simulation modelers concentrate on how to introduce the domain concepts and relations into a metamodel so as to enhance its semantic expressiveness. Specifically, the core of ontological metamodeling for semantic composability includes the following. Firstly, upon a set of domain specific modeling frameworks we introduce ontologies for those key elements that affect semantic composability and reuse much. Secondly, provide a formal and semantically accurate ontological modeling language as well as its supporting ontological modeling environment. Finally, engineer semantic composability within the MDE paradigm in order to develop simulation models in a composable way, and thus reduce the system development cost, promote the development effectiveness and enhance the quality of the final simulation products.

In general, different types of ontological metamodels for semantic composability are based on two different mechanisms: general-purpose ontology language like the OWL and UML profile for common ontologies [21]. Both of them have some similar advantages and disadvantages with ways (i.e., the implantation and tag ways) of building ontological metamodels. In the former case, simulation models involved in semantic composability are all specified by the OWL, that is, all of the simulation models conform to a similar ontological metamodel, i.e., the OWL metamodel. In the latter case, these simulation models have a common syntactical base, e.g., the UML, but their model elements lack accurate semantics for a special domain as well as a strong logic foundation. Both mechanisms are illustrated in Fig. 2.

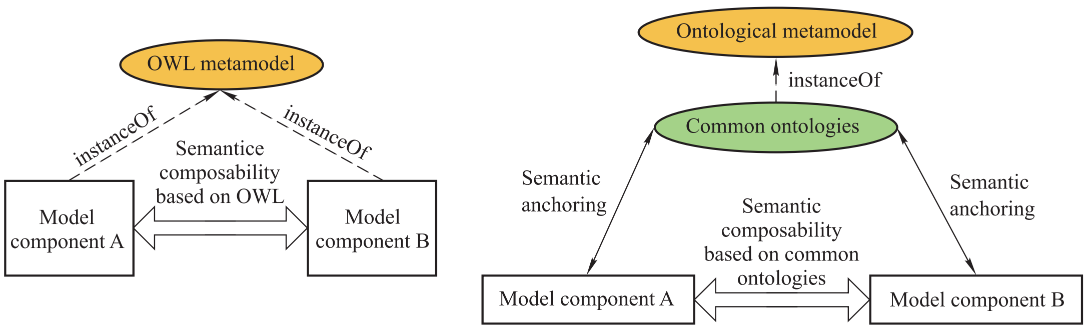

Fig. 2 Two mechanisms of semantic composability using ontological metamodels

3. Ontological metamodeling framework

Because of MDE's potential to dramatically change the way the simulation modelers develop simulation models, both practitioners and researchers have been working hard to deliver relative supporting technologies. Two well-known technologies that are commonly accepted and used in the MDE community are meta-object facility (MOF) and model-driven architecture (MDA), giving rise to two orthogonal directions of processes. One is concerned with language definitions from a vertical viewpoint, and the other is horizontally concerned with model transformations [22]. Instead of relegating either one to a secondary role, a well-defined framework for engineering semantic composability within MDE should give equal emphasis to both views, that is, neither should be subservient to the other [23,24].

Employing ontologies in MDE permits simulation modelers to group simulation models around ontologies [25,26]. In particular, the conceptual modeling stage plays a special role. Fig. 3 illustrates such an ontological metamodeling framework within MDE. In Fig. 3, CIM means computation independent model, PIM means platform independent model, PSM means platform specific model, CIM-MM means CIM-metamodel, PIM-MM means PIM-metamodel, and PSM-MM means PSM-metamodel. M2M means model to model, M2T means model to text, and ATL means atlas transformation language. This framework integrates the ontological metamodeling methodologies along with the MOF and the MDA, with the objective of engineering semantic composability of simulation models. The left side contains information about a given system from the perspective of system users. This side belongs to the problem domain, in which an analyzed method is often adopted and the models are usually represented in the form of descriptive models, e.g., the ontology. Whereas, the right side concerns details of a system's specification from the perspective of system developers. This side belongs to the methodology domain, in which a designed method is applied and simulation models are prescriptive models.

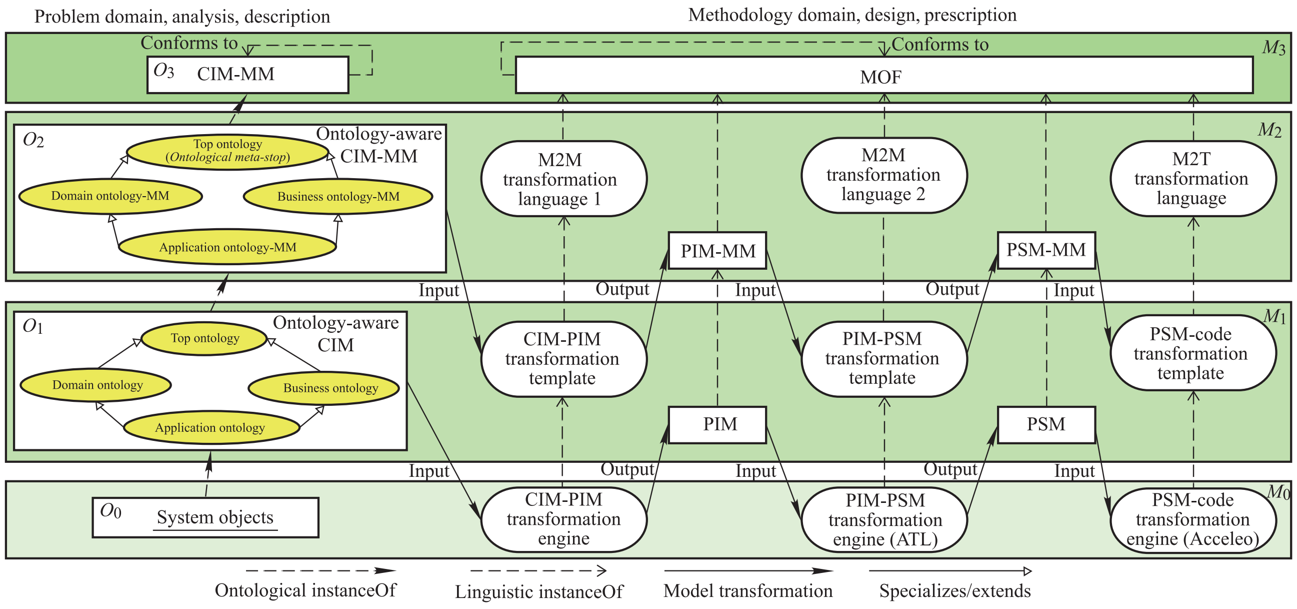

Fig. 3 Ontological metamodeling framework within MDE

The problem domain focuses on the description of real things, things that exist, unlike the artificial things that are designed to confine models in the methodology domain. We use the letter "O" plus a number "n" to notate the ontological meta-level on which a specific ontology lives, while the linguistic meta-level is notated by the letter "M" plus also a number "n". Hence, the ontologies on different ontological meta-levels are related with each other through the ontological instanceOf relation, while the relation between models on different linguistic meta-levels is linguistic instanceOf. Another two relations include "model transformation" that occurs either as an input or output, and "specializes/extends" that exists between ontologies on a similar ontological meta-level.

Ontologies that live on the meta-level O_0 correspond to system objects. An upper-level O_1 contains four practical types of ontologies discussed before: top ontology, domain ontology, business ontology and application ontology. Furthermore, the upper-level ontologies, also the standardized ones, live on O_2 since they provide a set of constructs for building O_1 ontologies. Then, the ontological instanceOf relation should stop on O_3 according to a "meta-stop principle" that is significant to terminate the unlimited meta-level of domain knowledge abstraction. This principle explains that, for a specific meta-level, if only there exists one single element or exclusive elements, the meta-level structure should stop intuitively at this level since there is no need to set a higher meta-level to abstract its commonalities. This is also the reason why the top ontology stops at O_2.

In the methodology domain, a collection of models are transformed in a clear and systematic way by defining a set of languages [27]. First of all, the CIM-PIM transformation engine takes the ontology-aware CIM as an input and outputs the PIM, and the CIM-PIM transformation template inputs the ontology-aware CIM-MM and outputs the PIM-MM. Secondly, PIM-MM is transformed into the PSM-MM by the PIM-PSM transformation template, and the PIM is transformed into the PSM by the PIM-PSM transformation engine. Finally, the PSM-MM is transformed into the code framework by the PSM-code transformation template. Also, the PSM is transformed into the final code by the PSM-code transformation engine. Hence, it is obvious that such an ontological metamodeling framework can benefit from ontologies, because via the standardized ontologies, the ontology-aware CIM requirements are able to be traceable to the PIM and the PSM constructs step by step, and vice versa.

In addition, this ontological metamodeling framework provides several other benefits. Firstly, employing ontologies into the CIM should increase the reliability of simulation products, because this ontology-aware CIM is going to be refined to the PIM and, then, via the PSM towards the final implementation. Secondly, ontologies provide a number of vocabularies that are readily captured and understood for system users, domain experts and simulation modelers. This should improve the communication between them. As a result, even domain experts can qualify the role of simulation modelers. Finally, refining models step by step should help enhance semantic composability of simulation models, because during the process of refinement, simulation modelers commence with a high level of abstraction and then refine and transform the models by adding more details of semantics, until they lead to a concrete application [28].

4. Building ontology-aware CMF

Indeed, it is not sufficient to adopt the MDE paradigm for engineering semantic composability of simulation models, neither is it sufficient to apply ontological metamodeling additionally, because ontological metamodeling is used to enhance the semantic expressiveness from the perspective of language definition. Rather, the key is to explore a well-defined mechanism to classify ever increasing simulation models. This mechanism should provide a common and effective management for different simulation resources, so as to precisely and rapidly locate a specific simulation resource. What is more, this mechanism should enable the technology of model abstraction to organize numerous simulation models and predefine the composable relations that inherently exist in them. With this mechanism, it is able to prescribe the efficient development of simulation models, and thus ensure the reuse of existing simulation resources. That is the ontology-aware composable modeling framework [29], namely ontoCMF.

4.1 What is ontoCMF?

The CMF elicits the commonalities that exist in various domain models to represent general knowledge concepts and relations but reserves their differences. As such, the abstract knowledge encapsulating in the CMF provides a unified support for the customization, management and evolvement of simulation models to acquire model reuse to a great degree. As a result, it is possible to develop simulation models in a composble way, and thus promote engineering simulation models. In fact, the ontoCMF is an abbreviation of the ontology-aware CMF in which the common information is elicited through ontological metamodeling. In this way, it benefits from not only the advantages of the CMF, but also the improvement of the understanding and interoperability of simulation models, because simulation models that use ontologies contain a common core of common vocabularies.

Similar to the CMF, the ontoCMF also has several features as follows. (i) Generality. It is able to define the structural and behavioral commonalities that are embedded in a lot of simulation models, and to prescribe the standard specifications of these simulation models. (ii) Constructiveness. It needs to construct a set of rules for semantic composability, explaining how to compose simulation models based on a unified CMF. (iii) Composability. Multiple simulation models can be composed into a valid and meaningful whole according to the constructive composable rules. (iv) Readability. It should be easily understood by simulation modelers, and has certain formal capabilities to be operated by computers. (v) Extendability. It should provide a predefined mechanism to extend simulation models under the change of outside environment or requirements.

The design of the ontoCMF mainly depends on two important modeling methodologies: knowledge abstraction and component-based development (CBD). The former one attempts to define a hierarchy of abstraction from both ontological and linguistic viewpoints. For example, the ontological hierarchy of a wire guided torpedo consists of tmSimModel, tmSimEntity, tmSimWeapon, tmTorpedo and tmWireGuidedTorpedo, while its linguistic hierarchy may contain metaclass, class and object. In this way, each layer only concerns the properties embedded in its layer. Moreover, a class that lives on a lower layer can reuse the properties of which lives on a higher layer. In general, the higher layer a class lives on, the more commonalities are embedded in this layer. Up to now, it is easy to find that the modeling task decreases a lot thanks to the knowledge abstraction. However, it is still not sufficient to support semantic composability of simulation models, so the latter methodology aims to classify the coupling relations that exist in various simulation models, uses mechanisms such as the aggregations, interfaces and events to capture them on an abstraction layer as high as possible, and decouples simulation models in order to develop them independently and to integrate them automatically. For example, the composable relationships between a platform and a weapon are predefined on a certain layer of abstraction, which could make a lot of prototypes of weapons that can be equipped in kinds of platforms, and a platform that has an access to weapons control.

4.2 Semantic composability of ontoCMF

So far, the studies to address model composability could mostly be summarized into two phases: standard specifications and semantics anchoring. The ways that are formed at the first phase mainly focus on the syntactical heterogeneity from a technological perspective, while the second phase attempts to address the semantic difference from a domain's perspective. Specifically, standard specifications aim to build a commonly accepted specification or formalism to represent models in a unified form. The semantics anchoring, whereas, intends to define good mapping relationships between different model elements for a particular domain. In practice, there exist two alternatives for mapping model elements [30,31], as shown in Fig. 4.

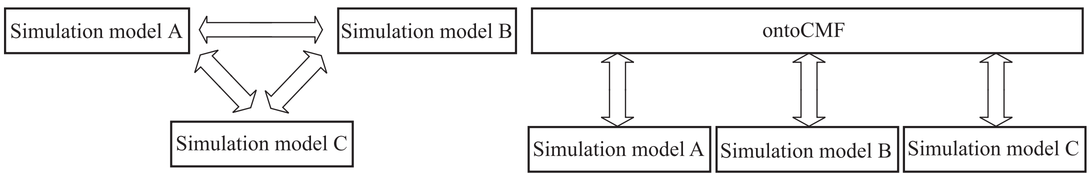

Fig. 4 Two alternatives for semantic mapping between simulation models

The left is to directly define a model to model mapping relationship between simulation models. This method does not intend to realize semantic composability of simulation models among all the domains, but limits to one or a couple of domains. Furthermore, it takes advantage of the inherent properties of simulation models in terms of reuse and composability to define their mapping relationships. For this reason, this method seems like a considerable pragmatic way in a small range of systems. Such typical systems generally have a long time of experience to be used in kinds of industries; thus, their simulation models gradually tend to be mature. However, it is still not easy to reuse their simulation models on a larger scale due to the relative closed system environment.

The right is to create a common modeling framework that is able to widely support a larger range of semantic composability. With this method, the common modeling framework functions as a bridge to indirectly connect different simulation models. In the context of MDE, the models have taken an increasingly important role during the development processes in system engineering. This trend has come along with the significant multiplication of modeling languages tailored to particular domains or applications, which incubates the inherent heterogeneity of simulation models. Therefore, researchers increasingly pay great attention to constructing a common model framework to support a wider range of semantic composability of simulation models.

4.3 OWL extension using UML profile

The OWL is a family of knowledge representation languages for authoring ontologies. According to different capabilities of expressiveness, the OWL family contains many species, such as OWL Lite, OWL description logic (DL) and OWL Full. Firstly, OWL Lite provides a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. Secondly, OWL DL is designed to provide the maximum possibility of expressiveness while retaining the computational completeness, decidability and the availability of practical reasoning algorithms. Thirdly, OWL Full allows an ontology to augment the meaning of the pre-defined vocabulary, but is not undecidable, so no reasoning software is able to perform complete reasoning for it.

Fig. 5 shows the classic OWL metamodel which

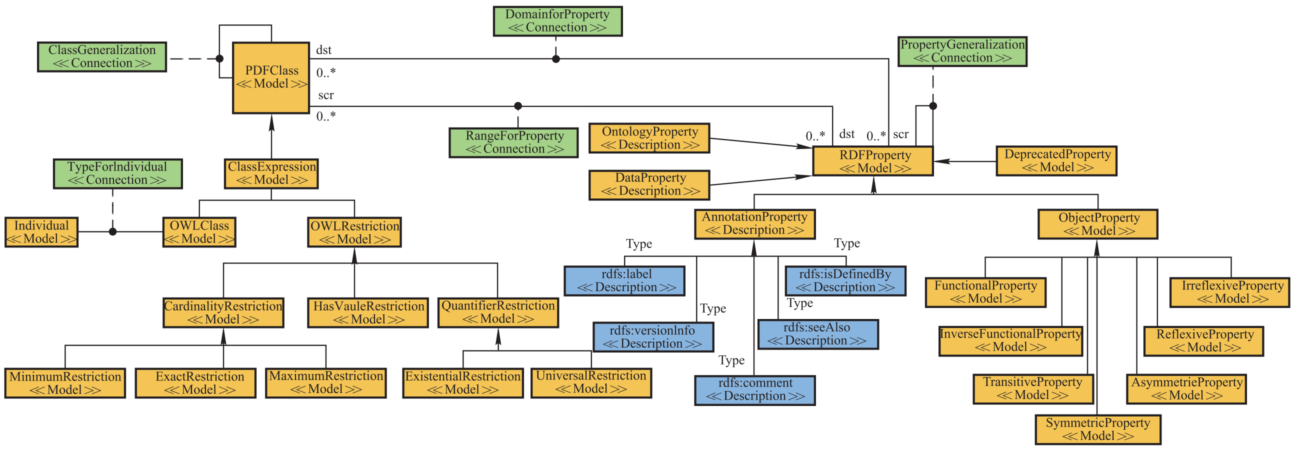
DomainforProperty ≪Connection≫
ClassGeneralization ≪Connection≫
PDFClass ≪Model≫
dst
0..*
scr
0..*
PropertyGeneralization ≪Connection≫
RangeForProperty ≪Connection≫
OntologyProperty ≪Description≫
DataProperty ≪Description≫
RDFProperty ≪Model≫
DeprecatedProperty ≪Model≫
TypeForIndividual ≪Connection≫
ClassExpression ≪Model≫
Individual ≪Model≫
OWLClass ≪Model≫
OWLRestriction ≪Model≫
AnnotationProperty ≪Description≫
ObjectProperty ≪Model≫
Type
rdfs:label ≪Description≫
rdfs:isDefinedBy ≪Description≫
rdfs:versionInfo ≪Description≫
rdfs:seeAlso ≪Description≫
rdfs:comment ≪Description≫
CardinalityRestriction ≪Model≫
HasVauleRestriction ≪Model≫
QuantifierRestriction ≪Model≫
MinimumRestriction ≪Model≫
ExactRestriction ≪Model≫
MaximumRestriction ≪Model≫
ExistentialRestriction ≪Model≫
UniversalRestriction ≪Model≫
FunctionalProperty ≪Model≫
IrreflexiveProperty ≪Model≫
InverseFunctionalProperty ≪Model≫
ReflexiveProperty ≪Model≫
TransitiveProperty ≪Model≫
AsymmetrieProperty ≪Model≫
SymmetricProperty ≪Model≫

Fig. 5 Classic OWL metamodel

contains three core elements, i.e., Individual, Property, and Class. To begin with, Property has many types such as OntologyProperty, DataProperty, AnnotationProperty, ObjectProperty and DeprecatedProperty. AnnotationProperty can be typed by Label, versionInfo, comment, seeAlso and isDefinedBy. ObjectProperty is further inherited by FunctionalProperty, InverseFunctionalProperty, TransitionProperty, SemetricProperty and AsmmetricProperty. Then, one can define a special class by adding a Restriction to an ObjectProperty. Restriction consists of Cardinality Restriction, HasValue and Quantifier. First, Cardinality Restriction describes the minimum, maximum, or a special value relation that exists between two different classes. Second, HasValue describes the relation that exists between an individual and another individual through the ObjectProperty connection. Third, QuantifierRestriction is divided into Existential and Universal. The former refers to a special relation of an individual with at least one individual of another class, while the latter refers to a special relation of an individual with only one individual of another class. In addition, Class is associated with Individual through TypeForIndividual, and has a many-to-many relation with Property. The classes connected by a special Property are called Domain and Range of this Property. Either Class or Property has its own generation relation, namely Class GeneralizationorProperty Generalization.

Table 1 presents how the OWL Lite elements are mapped to the UML metaclasses, and also their corresponding UML profile mechanisms [32], as well as the mathematical semantics. Aiming at a set of basic modeling elements of OWL Lite, the UML profile provides stereotype, tagged values and constraints to extend the fundamental UML metaclasses such as Association, Class, Package, Enumeration, Literal, Multiplicity and Inheritance relation, and thus the semantic expressiveness of part of UML metaclasses increases much as a result. We call this profile ontoUML, and its ontological modeling environment ontoUMLTool will be discussed later.

Table 1 UML profile for OWL Lite (part)

OWL Lite	UML	UML profile	Mathematical semantics
Symmetric	Literal	Tagged values	$Po_1 \rightarrow Po_2 \Rightarrow Po_2 \rightarrow Po_1$
Asymmetric	Literal	Tagged values	$Po_1 \rightarrow Po_2 \Rightarrow \neg(Po_2 \rightarrow Po_1)$
Reflexive	Literal	Tagged values	$Po_1 \in Po_2 \in Po_3, \ldots, \in Po_n$
Irreflexive	Literal	Tagged values	$\neg(Po_1 \in Po_2 \in Po_3, \ldots, \in Po_n)$
maxCardinality	Multiplicity	<UML2 kernel>	$\leqslant n.P$
minCardinality	Multiplicity	<UML2 kernel>	$\geqslant n.P$
exactCardinality	Multiplicity	<UML2 kernel>	$= n$
HasValue Restriction	Association	Stereotype	$\exists P$
UnionOfRelation	Association	Stereotype	$R_1 \cup \cdots \cup R_n$
NonUnionOfRelation	Association	Stereotype	$\neg(R_1 \cup \cdots \cup R_n)$

4.4 Ontological metamodeling environment: ontoUMLTool

The OWL has a widely used tool support named Protégé [33], which is able to satisfy the implantation way to build ontological metamodels. However, we often need to construct a new tool for the tag way to build ontological metamodels. In fact, there exists not a unified standard because different simulation modelers may have different preferences and also the information that is needed to be tagged is often different, leading to a tremendous difference of ontological metamodels. Based on theUML profile, we develop a tool called ontoUMLTool in support of the profile ontoUML, as shown in Fig. 6.

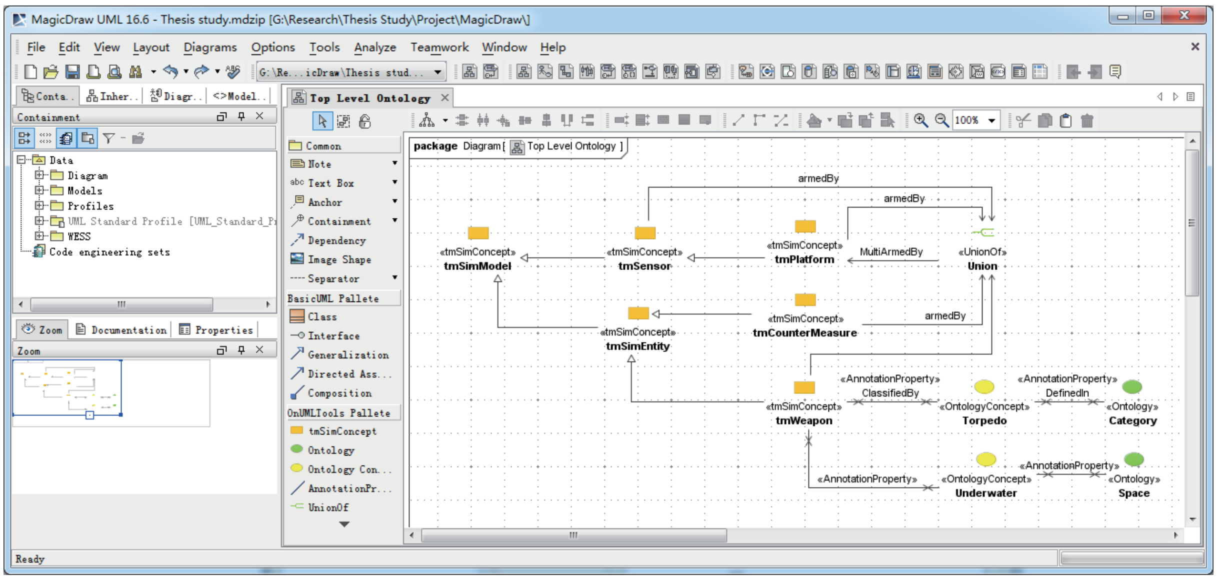

Fig. 6 Ontological metamodeling environment within MagicDraw

Within the ontoUMLTool, we can apply the tag way to build a top level ontology in the editing area. The top level ontology describes a set of domain specific concepts, relationships and tags for the CESS domain. First of all, tmSimModel as the most fundamental concept is inherited by tmSensor and tmSimEntity. Note tmSimEntity differs from tmSensor in that the former has a motion concept but the latter has not. Furthermore, tmSimEntity derives tmWeapon, tmCounterMeasure and tmPlatform. Secondly, it is possible for tmPlatfrom to be armed by a union of tmSensor, tmCounterMeasure, tmWeapon and another sub tmPlatfrom. Finally, tmWeapon can be designated as a Torpedo according to the ontology Category and has a property named Underwater according to the Space ontology.

In addition, the tool palette of the ontoUMLTool contains three main blocks, i.e., Common, BasicUML Palette and ontoUMLTools Palette. The first block is concerned with several general decorations such as Note, Text Box and Anchor, and the second holds a collection of modeling elements of the UML class diagram. For the third block, it contains a set of domain specific concepts which simulation modelers can customize according to their special requirements, such as ≪tmSimConcept≫, ≪Ontology≫, and so forth.

5. Case study: CESS

In recent years, CESS is becoming one of the most important supporting means to combat systems acquisition and conceptual design. Plenty of research has been carried out and a number of CESS systems are developed as a result. This section describes an example application of the CESS engineering process to illustrate the semantic composability of simulation models based on ontological metamodeling. Initially, we propose a three-decomposition plus a two-layer knowledge architecture to reduce the complexity of CESS specification, and continue designing a set of CESS ontoCMFs, and then develop an illustrative example of underwater targets search to demonstrate the final simulation implementations.

5.1 CESS ontoCMFs

As shown in Fig. 7, a set of ontologies such as the top level, domain, behavior, process, application, and simulation scheduling ontoCMFs are also identified as same as the hierarchical structure of ontology [34]. These ontoCMFs are layered with three levels that are notated by the letter “F” plus a number “n”. Each level refers to a specialized or extended level on which a specific ontoCMF lives.

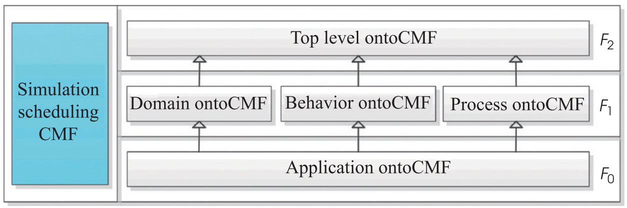

Fig. 7 Hierarchical structure of ontoCMF

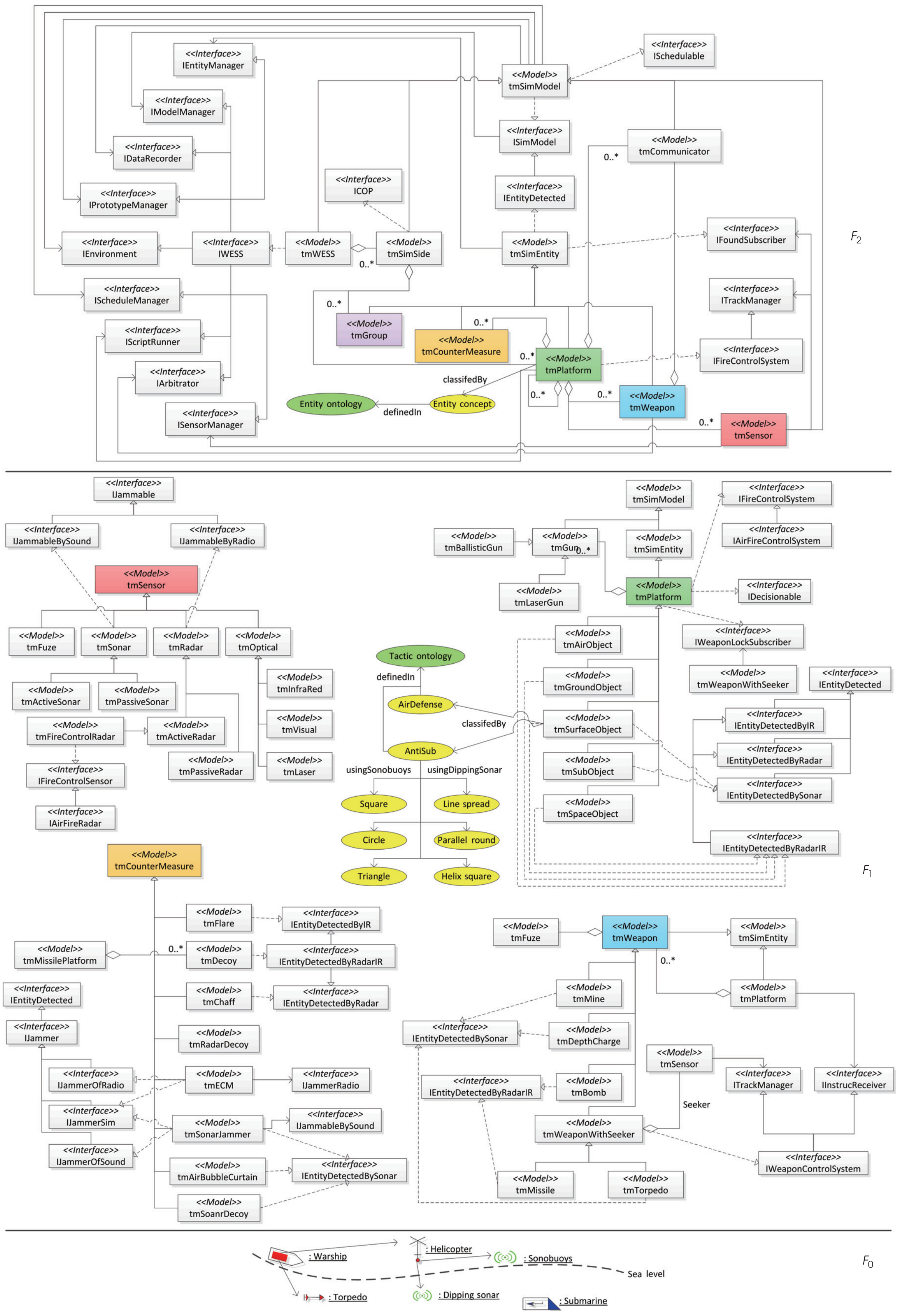

Fig. 8 A set of CESS ontoCMFs

Note that the simulation scheduling CMF is a key technology in support of simulation running, which belongs to the range of simulation engine and has nothing to do with the special application or domain.

Firstly, the top level ontoCMF on F_2 is the core of the overall ontoCMFs. In the form of abstraction, it uses the most fundamental concepts and relationships to restrict the range that is able to reach when describing ontoCMFs on the levels below. Secondly, a lower level F_1 specializes or extends some specific elements of ontoCMFs on F_2 to build domain, behavior, and process ontoCMFs. First, the domain ontoCMF is concerned with the static entities, properties, operations, as well as the inheritance, dependence, and aggregation relations. Hence, it is mainly described by the UML class diagram or the entity-relationships (ER) diagram. Second, the behavior ontoCMF is concerned with the states, transitions and events and hence the finite state machine (FSM) and the dynamical model are widely applied. Third, the process ontoCMF is concerned with the tasks, activities, and the procedure on which these activities are executed. The integrated definition methods (IDEF) and the business process modeling notation (BPMN) are usually used to describe them. Finally, the level F_0 describes the most concrete simulation entities, relationships and various data resources involved in a special scenario.

Fig. 8 shows several typical CESS ontoCMFs. Based on the original CMFs that are described as UML class diagrams, we adopt the second ontological metamodeling method, i.e., UML extension with tags, to build these ontoCMFs for the representation of tactics. To begin with, on the level F_2 is the top ontoCMF, which consists of the most fundamental concepts and relations of the CESS. We select tmGroup, tmCounterMeasure, tmPlatform, tmWeapon and tmSensor as the extension points to construct domain ontoCMFs on the level F_1. Secondly, the level F_1 contains a set of domain ontoCMFs with respect to the weapon, platform, countermeasure and sensor, and note the ontoCMF for group is omitted for brevity. In the ontoCMF for platform, tmPlatform is tagged by two kinds of tactics: AirDefense and AntiSub. In fact, there are more tactics that could be tagged on this level. For brevity, here we only present relative tactics according to the concrete scenario on the level F_0. These details will be discussed later. Finally, on the level F_0 there is a concrete scenario of engagement between two sides, where the red side is an anti-submarine group that consists of a warship and a helicopter, while the blue side contains only an enemy submarine.

5.2 Simulation results and analysis

The weapon effectiveness simulation system (WESS) is an M&S platform for the CESS domain [35]. In general, the WESS has two workflows from different perspectives: domain model development (DMD) for modelers and simulation application development (SAD) for users, and the SAD starts when the DMD finishes. In the SAD workflow, a set of tools are developed to support collecting data and editing scenarios, designing experiments and tactics, and displaying and analyzing simulation results.

The purpose of scenario editing is to deploy the forces of two sides and their initial status, as well as to configure the attributes of each entity. The configuration contains four things: setting up the prototype and tactical data of each entity, associating the decision script with each platform entity, adding weapons and sensors for each platform entity, and planning the initial waypoints of each platform entity. In this scenario, we assume that an adversary will cruise through an ocean area at a certain velocity, and its acoustic signature is already detected by the ground-based sonar, or by other information means. Thus, the warship, cooperatively equipped with a helicopter, is ordered to conduct the mission of searching the adversary submarine. The helicopter has two devices to conduct this mission. For the sonobuoys, we just need to set the number of sonobuoys in the panel of attributes configuration. For the dipping sonar, we need to set the number of dipping points in the decision script.

When all the data are configured completely and simulation models are prepared, the next work is to run simulations and analyze the results to get meaningful information. We set the total logical running time for each simulation to 1 000 s and perform 500 rounds of Monte Carlo simulations. Fig. 9 shows the two-dimension (2D) simulation display of each tactic. A benefit of the simulation display is that it assists us in a visual way to grasp the whole real-time situation of the battlefield, releasing us from thousands of lines of codes or texts. Furthermore, we just need to change the value of any decision parameter in the same scenario, not to construct a new scenario from scratch, and thus a different search pattern can be acquired. For example, a triangle search pattern is easily acquired by an assignment of searchMode = C_searchTriangle which is originally the assignment of searchMode = C_searchCircle. In addition, taking Fig. 9(e) as an example, we can see the circle consists of 12 sonobuoys, and each pair of two adjacent sonobuoys connects to form a polygon. If an adversary submarine acts inside the circle, then it is not easy to escape from any range of this search pattern.

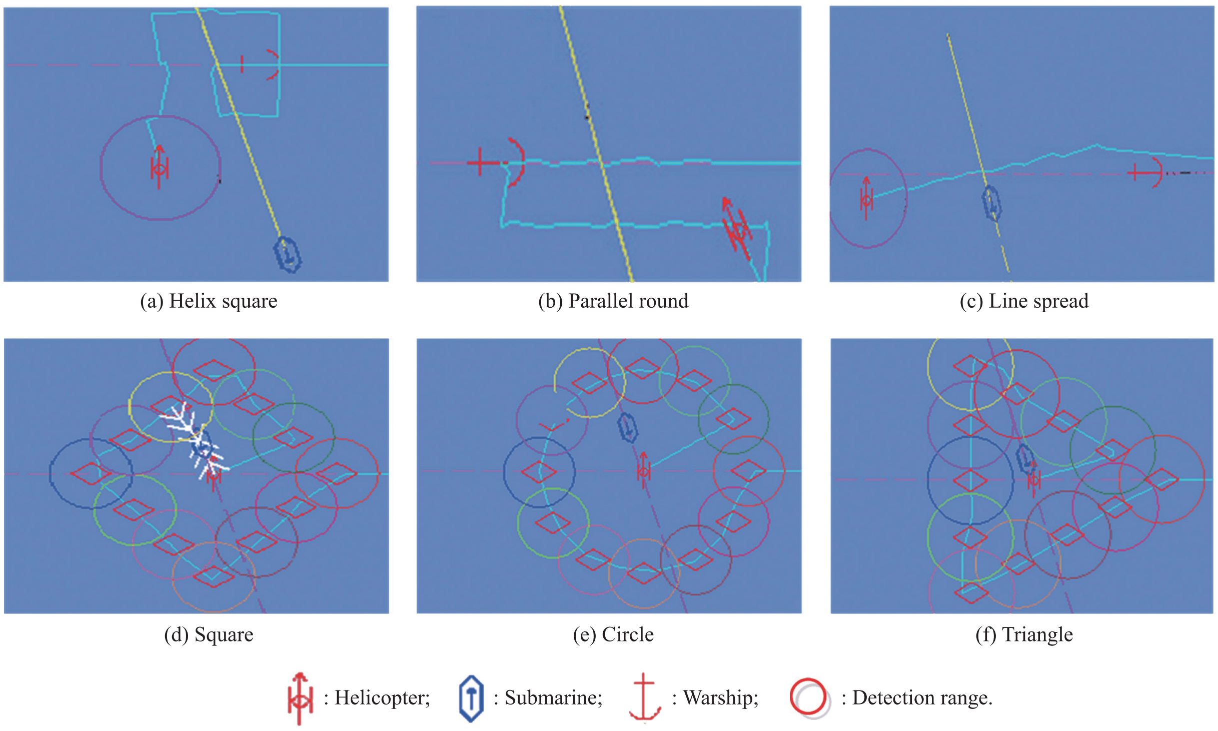

Fig. 9 2D simulation display of each tactic

6. Conclusions

Abstraction is now widely admitted as an effective means to reduce the complexity of system specification. It is also generally agreed that the ontology as an important form of abstraction can be employed in MDE to describe the existing world, the environment and the domain of a system. However, this consensus has not lead to a coherent research on how to enhance the semantic composability of simulation models yet. Hence, this study attempts to adopt an ontological metamodeling method for engineering the semantic composability of simulation models within MDE. We believe that the experience collected from this study can bring some new visions of simulation models development and semantic composability.

This study introduces the basic concepts of model, ontology, and metamodel initially, then moves forward to conclude the ontological metamodel concept and also identifies two general ways to build an ontological metamodel. Correspondingly, we present two typical mechanisms for semantic composability based on ontological metamodeling, in which one corresponds to the implantation way and the other to the tag way. After that, the ontological metamodeling framework is proposed. This framework emphasizes the role of ontologies at the phase of conceptual modeling with the objective to define a set of ontology-aware CMFs, namely ontoCMFs. Then, we give what an ontoCMF is and discuss two widely used alternatives to realize semantic conposability. As a proof of concept, the CESS is adopted to design a set of ontoCMFs, and an illustrative simulation example of underwater targets search is simulated. However, as a drawback more simulations about different scenarios are necessarily required as further illustrations.

References

[1] ZEIGLER B P. How can modeling and simulation help engineering of system of systems? Amsterdam, the Netherlands: Elsevier, 2017.

[2] SARJOUGHIAN H S. Model composability. Proc. of the 38th Conference on Winter Simulation, 2006: 149 – 158.

[3] SZABO C, TEO Y M. On syntactic composability and model reuse. Proc. of the 1st Asia International Conference on Modeling and Simulation, 2007: 230 – 237.

[4] SZABO C, TEO Y M. An analysis of the cost of validating semantic composability. Journal of Simulation, 2012, 6(3): 152 – 163.

[5] IEEE Computer Society. IEEE 1516-2010-standard for modeling and simulation high level architecture-framework and rules. http://www.ieee.org/HLA.

[6] European Space Agency. SMP 2.0 handbook. http://www.eurosim.nl/support/manuals/manual_4_2/pdf/SMP_2.0_Metamodel-1.2.pdf.

[7] ZEIGLER B P, PRAEHOFER H, KIM T G. Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems. 2nd ed. New York: Academic Press, 2000.

[8] TENDELOO Y V, VANGHELUWE H S. Classic DEVS modelling and simulation. Proc. of the Winter Simulation Conference, 2017: 644 – 658.

[9] AZAR M C. Assessing the treatment of airborne tactical high

energy lasers in combat simulations. Dayton, U.S.A.: Air Force Institute of Technology, 2003.
[10] MILLER J O, JASON L, HONABARGER B. Modeling and measuring network centric warfare (NCW) with the system effectiveness analysis simulation (SEAS). Proc. of the 11th International Command and Control Research & Technology Symposium, 2006: 1 – 22.
[11] HALL S B, ZEIGLER B P, SARJOUGHIAN H S. Joint measure TM: distributed simulation issues in a mission effectiveness analytic simulator. Proc. of the Simulation Interoperability Workshop, 1999: 1 – 7.
[12] LEI Y L, LI Q, YANG F, et al. A composable modeling framework for weapon systems effectiveness simulation. Systems Engineering – Theory & Practice, 2013, 33(11): 2954 – 2966. (in Chinese)
[13] PIDD M. Tools for thinking-modeling in management science. New York: Wiley, 2009.
[14] Object Management Group. MDA guide. http://www.omg.org/mda.
[15] SEIDEWITZ E D. What models mean? IEEE Software, 2003, 20(5): 26 – 32.
[16] FAVRE J M, NGUYEN T. Towards a megamodel to model software evolution through transformations. Electronic Notes in Theoretical Computer Science, 2005, 127(3): 59 – 74.
[17] AßMANN U, ZSCHALER S, WANNER G. Ontlogies, metamodels, and model-driven paradigm. New York: Springer Verlag, 2006.
[18] SELIC B. A systematic approach to domain-specific language design using UML. Proc. of the 10th IEEE Internatinal Symposium on Object and Component-oriented Real-time Distributed Computing, 2007: 2 – 9.
[19] NORDSTROM G, SZTIPANOVITS J, KARSAI G, et al. Metamodeling – rapid design and evolution of domain-specific modeling environments. Proc. of IEEE Conference on Engineering of Computer-based Systems, 1999: 68 – 74.
[20] LEI Y L, ZHU N, YAO J, et al. Model-architecture oriented combat system effectiveness simulation. Proc. of the Winter Simulation Conference, 2015: 3190 – 3191.
[21] HAPPEL H J, KORTHAUS A, SEEDORF S, et al. KontoR: an ontology-enabled approach to software reuse. Proc. of the 18th International Conference on Software Engineering & Knowledge Engineering, 2006: 349 – 354.
[22] ÇETINKAYA D. Model driven development of simulation models: defining and transforming conceptual models into simulation models by using metamodels and model transformations. Ankara, Turkey: Middle East Technical University, 2003.
[23] HITZ M, KESSEL T. Using application ontologies for the automatic generation of user interfaces for dialog-based applications. Proc. of Research and Practical Issues of Enterprise Information Systems, 2016: 16 – 31.
[24] TUENO S, LALEAU R, MAMMAR A, et al. Towards using ontologies for domain modeling within the SysML/KAOS approach. Proc. of the 25th International Requirements Engineering Conference Workshops, 2017: 1 – 5.
[25] ATKINSON C, KUHNE T. Model-driven development: a metamodeling foundation. IEEE Software, 2003, 20(5): 36 – 41.
[26] ZHU Z, LEI Y L, SARJOUGHIAN H S, et al. UML-based combat effectiveness simulationsystem modeling within MDE. Journal of Systems Engineering and Electronics, 2018, 29(6): 1180 – 1196.
[27] LACKO P, KAJSA P, NAVRAT P. Design pattern instances within model driven development based on abstraction, concretization and variability. Computing and Informatics, 2017, 36(1): 55 – 85.
[28] ZHU Z, LEI Y L, LI Q, et al. Exploring MDE techniques for engineering simulation models. Wireless Networks, 2020, 26(4): 1 – 12.
[29] ZHU Z, LEI Y L, ZHU N, et al. Composable modeling frameworks for networked air & missile defense systems. Journal of National University of Defense Technology, 2014, 36(5): 186 – 190. (in Chinese)
[30] SMIRNOV P A, KOVALCHUK S V, DUKHANOV A V. Domain ontologies integration for virtual modelling and simulation environments. Procedia Computer Science, 2014, 29(1): 2507 – 2514.
[31] BENJAMIN P, AKELLA K, VERMA A. Using ontologies for simulation integration. Proc. of the Winter Simulation Conference, 2007: 1081 – 1089.
[32] ABDULAH M S. A UML profile for conceptual modeling of knowledge-based systems. York, U.K.: University of York, 2006.
[33] Protégé. Plugin Anatomy. http://protegewiki.stanford.edu/wiki/PluginAnatomy.
[34] ZHU Z, LEI Y L, ZHU Y F. Model-driven combat effectiveness simulation systems engineering. Defence Science Journal, 2020, 70(1): 54 – 59.
[35] LEI Y L, ZHU Z, LI Q, et al. WESS: a generic combat effectiveness simulation system. Proc. of the 17th Asia Simulation Conference, 2017: 272 – 283.

Biographies

LEI Yonglin was born in 1978. He received his Ph.D. degree in management science and engineering from National University of Defense Technology in 2006. He is now a professor at National University of Defense Technology. He was also a visiting scholar at Arizona State University. His research interests are complex system modeling & simulation, model-driven engineering and simulation composability.
E-mail: yllei@nudt.edu.cn

ZHU Zhi was born in 1989. He received his M.S. degree in control science and engineering and Ph.D. degree in military equipment from National University of Defense Technology in 2013 and 2018, respectively. He is now an assistant professor at National University of Defense Technology. He was also a visiting Ph.D. student at Arizona State University. His research interests are language-driven development and system engineering.
E-mail: zhuzhi@nudt.edu.cn

LI Qun was born in 1971. He received his M.S. and Ph.D. degrees in management science and engineering from National University of Defense Technology in 1995 and 1997, respectively. He is now a professor at National University of Defense Technology. He was also a visiting scholar at Nanyang Technological University. His research interests are software engineering, simulation-based design and agent-based simulation.
E-mail: liqun@nudt.edu.cn