Cognitive twin construction for system of systems operation based on semantic integration and high-level architecture

Article *in* Integrated Computer Aided Engineering · March 2022

DOI: 10.3233/ICA-220677

Integrated Computer-Aided Engineering 29 (2022) 277–295
DOI 10.3233/ICA-220677
IOS Press

Cognitive twin construction for system of systems operation based on semantic integration and high-level architecture

Han Li[a], Guoxin Wang[a], Jinzhi Lu[b,*] and Dimitris Kiritsis[b]
[a]*School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China*
[b]*SCI-STI-DK, EPFL, CH-1015 Lausanne, Switzerland*

Abstract. With the increasing complexity of engineered systems, digital twins (DTs) have been widely used to support integrated modeling, simulation, and decision-making of the system of systems (SoS). However, when integrating DTs of each constituent system, it is challenging to implement complexity management, interface definition, and service integration across DTs. This study proposes a new concept called cognitive twin (CT) to support SoS development and operation. CTs have been defined as DTs with augmented semantic capabilities for promoting the understanding of interrelationships be-tween virtual models and enhancing the decision-making. First, CTs aim to integrate the information description of DTs across constituent systems using a unified ontology and semantic modeling technique. Second, CTs provide integrated simulations among DTs for decision-making of the SoS based on a high-level architecture (HLA). Finally, through reasoning ontology models, CTs provide decision-making options for the operations of real constituent systems. A case study on unmanned aerial vehicles (UAVs) landing on unmanned surface vehicles (USVs) is used to verify the flexibility of this approach. From the results, we find that the CT based on the proposed ontology provides a unified formalism of DTs across UAVs and USVs. Moreover, the reasoning based on the CT provides decision-making capabilities for UAVs by implementing cognitive computing to select target USVs for landing.

Keywords: System of systems, model-based systems engineering, ontology, cognitive twin, high-level architecture

1. Introduction

Developers of complex systems face an emergent challenge in the integration of heterogeneous complex systems across domains [1,2]. Currently, systems that are integrated within an infrastructure by implementing internet connectivity between physical and cyber devices cannot exist in isolation, which drives widespread and accelerating changes in system environments [3]. Thus, a system of systems (SoS) has been proposed to define such a situation as a "set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own" [4], such as an air traffic control system [5]. Moreover, to support SoS analysis and development, system of systems engineering (SoSE) has been proposed as "the process of planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a system-of-systems capability that is greater than the sum of the capabilities of the constituent parts" [6].

When developing an SoS and its constituent systems, complexity is a key performance indicator to measure and manage the interrelationships among constituent systems and their components. Complexity refers to a system characteristic that makes it difficult, or even impossible, to accurately predict behavior over time, particularly in terms of understanding all relevant interactions among system components within the defined system boundary [7]. When managing complexity, the unified formalism of the SoS is the basis to describe the topology between each system and component [8].

*Corresponding author: Jinzhi Lu, SCI-STI-DK, EPFL, CH-1015 Lausanne, Switzerland. E-mail: jinzhi.lu@epfl.ch.

However, discrepancies of system feature within one SoS require a unified specification to represent domain-specific knowledge of each constituent system, which is herein considered as *challenge 1*.

Digital twins (DTs) have been widely used to provide performance predictions for constituent systems as a virtual representation that serves as a real-time digital counterpart of a physical object or process [9]. Within an SoS, it is common to involve several constituent systems with their respective DTs for the system development. However, such heterogeneous DTs cannot be integrated to predict the performance of SoS operations because of the various data structures and interface specifications. Moreover, interface management when integrating such DTs is a difficult task, because of the complex topology among the DTs. Finally, when implementing the integrated simulation across DTs, run-time and synchronized communications are required to support data exchange across DTs. Thus, a standardized framework is needed to combine DTs for integrated simulation across an SoS, which is herein considered as *challenge 2*.

Because an SoS involves different constituent systems, gaps between the development processes of all the constituent systems lead to uncertainties when implementing such systems together. In this situation, DTs enable the implementation of real-time communications between the physical and virtual worlds to make decisions for the real constituent systems during their operational processes [10]. However, it is difficult to realize this mechanism using current hardware and software, because of the limitation of real-time computing capabilities. Thus, a viable approach is expected to provide decision-making options for the behaviors of constituent systems during SoS operation based on simulation results, which is herein considered as *challenge 3*.

To address these challenges, a cognitive twin (CT) concept is proposed to encompass complexity management of SoSs and their DTs, to implement integrated simulation, and to support decision-making for the SoS operations. In this approach, a unified ontology is developed to support semantic modeling for describing the topology among SoSs and their DTs. Moreover, integrated simulation is used to predict the behaviors of constituent systems using DTs. Based on the simulation results, reasoning of semantic models is implemented to make decisions during SoS operations.

The CT aims to provide a semantic modeling approach to formalize an SoS using a unified description. Moreover, high-level architecture (HLA) [11] simulations enable CTs to provide an integrated simulation environment for SoS analysis. Finally, simulation results are the basis to support decision-making for SoS operations based on semantic reasoning. Compared with traditioal SoS development, CT provides the formalization of virtual entities, SoS, and integrated simulation results. The contributions of this paper are as follows:

- Support for complexity management of an SoS by defining the topology among constituent systems within the SoS and their DTs: The Basic Formal Ontology (BFO) [12] and Industrial Ontologies Foundry (IOF) [13] are referenced to define the ontology for semantic modeling of an SoS and the related DTs. The unified ontology enables the description of real constituent systems and their DTs for complexity management by transforming non-structural complexity into transparent structural complexity.
- Support for managing the interfaces between heterogeneous DTs in an integrated simulation infrastructure. SoS development involves heterogeneous DTs for each constituent system; thus, HLA is required to integrate DTs for behavior prediction.
- Support for decision-making of system behaviors through reasoning and integrated simulation: Based on the integrated simulation results, semantic models of the SoS and DTs are extended with the simulation results referring to constituent system behaviors. The reasoning of such models enables decision-making for SoS operation.

The remainder of this paper is organized as follows. We discuss the related work in Section 2 and demonstrate our research methodology in Section 3. In Section 4, we specifically introduce a semantic modeling approach, including ontology design, integrated simulation across DTs for SoS development, and semantic integration. In Section 5, we present a case study to clarify and validate the proposed CT concept. Finally, discussions are presented in Section 6, and the conclusions of this work are drawn in Section 7.

2. Related work

In this section, we first identify the challenges faced by systems engineers from the perspective of SoSs. Then, we introduce DTs and their integration based on the existing research. Moreover, semantic modeling is investigated to support SoS development. Finally, the state of the art of CTs is discussed to summarize the motivations of this study.

2.1. System of systems challenges

SoSs are currently of great interest to systems engineers [14]. SoSE was proposed as a specific systems engineering approach to manage SoS development [15]. Currently, SoSE is facing several challenges [16]: 1) *Ambiguity description across SoS*: it leads to a large number of design errors and increased project cycle time and cost; 2) *Architecture design and verification across systems*: SoS is composed of many heterogeneous systems which is designed and verified its respective system developers. This leads to a challenge in terms of interoperability across domain-specific knowledge; 3) *Assessing the behaviors in real-time SoS situations*: an SoS involves different constituent systems, and their design gaps lead to uncertainties during SoS operations, which require flexible decision-making to decrease the introduced risks. Thus, autonomous decision-making for SoS operations is useful to provide more flexible behaviors for each system.

2.2. Digital twin integration for system of systems

The concept of a DT originated from Grieves's speech on product lifecycle management in 2003 [17]. DTs should at least contain three basic elements: physical entities, virtual entities, and communications between them through data and information. The virtual entities of DTs are considered as digital models, referring to a virtual representation that contains all physical information and knowledge [18]. Currently, DTs are widely used to support SoS development [19]. DTs are developed for different constituent systems during development processes, which makes the integration of heterogeneous data difficult [20]. Several techniques have been proposed to support DT integration within an SoS. For example, HLA is a standard for the modeling and co-simulation of distributed processes and provides standardized interface specification which makes it possible to integrate different hardware interfaces into a single SoS [21,22]. Moreover, distributed co-simulation protocol (DCP) is also used to support distributed simulation across DTs to enable integrated verification of SoSs [23], but it mainly addresses the integration problem of real-time simulation.

2.3. Semantic modeling for system of systems

Semantic modeling enables computers to understand human knowledge, and it has been widely used to describe domain-specific knowledge [24,25]. Moreover, it provides solutions for data integration of DTs using web techniques [26]. Metamodeling methods provide the basis for semantic modeling. Graph, Object, Point, Property, Role, Relationship (GOPPRR) is the metamodeling language that can express almost all complex relationships. In addition, the Generic Modeling Environment (GME) is also a metamodeling method used in the area of electrical engineering [27]. To date, existing studies have attempted to integrate semantic models with DTs through ontology [28]. For example, Unified Modeling Language(UML) and ontology were integrated to create a high-level semantic model for the exchange of information between computers and humans through human-readable text and computer-readable models [29,30].

2.4. State of the art of cognitive twins

Previous researchers explored to enhance DTs with cognitive capabilities using semantic technologies and DTs. Semantic modelling and ontologies were proposed as a concept of semantically enhanced DTs which enables to define system characteristics, as well as the topology that it interacts with other components [31]. The knowledge graph was used to connect and retrieve heterogeneous data including descriptive and simulation models, it was proposed as the paradigm of the next generation DTs [32]. In 2016, a workshop presentation from Ahmed El Adl was presented about the cognitive evolution of IoT technologies who proposed a Cognitive Digital Twin concept. It was defined as *"a digital representation, augmentation and intelligent companion of its physical twin as a whole, including its subsystems across all of its life cycles and evolution phases"*. Recently, the Cognitive twins (CTs) were formally proposed as *"DTs with augmented semantic capabilities for identifying the dynamics of virtual model evolution, promoting the understanding of interrelationships between virtual models and enhancing the decision-making"* [33]. With the similar concept, the functions of hybrid human-machine cognitive ssystems were investigated with the CT definition as *"a digital expert or copilot, which can learn and evolve, and that integrates different sources of information for the considered purpose"*. From the technical perspective, CT has its framework to combine semantic modeling and integrated simulation in order to support SoS development.

2.5. Summary

From the literature review, we find several key motivations for this study:

– The SoS is an emergent concept of interest to system developers. SoSE has been proposed to support and manage SoS development, including methods, software, and simulations. Currently, ambiguous descriptions across an SoS, assessing SoS behaviors, and autonomous decision-making for SoS operations are three challenges within the SoSE domain.

- Integration of DTs is challenging because of heterogeneous data and model structures during SoS development. Currently, HLA, Distributed Co-Simulation Protocol (DCP), etc., are widely used to support DT and model integration. Such techniques are effective in supporting data communication when integrating DTs.
- Semantic modeling is important for constructing DTs; however, it still lacks a unified architecture and cannot solve the challenge of integrating different domain-specific knowledge.
- The CT is a new concept for supporting SoS development by using semantic modeling and integrated simulation. Based on this concept, a new potential solution enables managing the SoS complexity through unified descriptions of constituent systems and their topology. Moreover, it uses integrated simulation results to support decision-making for SoS operations.

3. Research methodology

In this section, the research methodology is introduced to develop and evaluate the proposed CT concept for SoS development.

3.1. Construction of cognitive twins based on systems thinking and reference ontology

Figure 1 illustrates the research design. Systems thinking is first used to capture the entities and their topology, which represent the real SoS, and their DTs. Such entities and topology represent domain knowledge about the SoS to construct DTs for each constituent system. These DTs are combined based on the domain knowledge to implement integrated simulation and to obtain the results for each constituent system behavior.

Then CT ontology is defined to contain all entities in the scope, including physical entities of SoS, virtual entities and the topology among them, and simulation results. IOF model-based systems engineering(MBSE) ontology is used to formalize the virtual entities of the constituent systems in the SoS. The IOF MBSE ontology is based on meta – meta models consisting of six key concepts with extensions: Graph, Object, Point, Property, Role, and Relationship(GOPPRRE) [34,35]. Reference ontologies including BFO [12] and IOF specifications [13] are used to formalize the domain knowledge related to the SoS, DTs, and simulation results. Then, the ontology is aligned to the BFO specification for the final CT ontology. The CT ontology, DTs, and SoS are used to construct CTs that provide ontology models for reasoning. The reasoning results support decision-making for manipulating the constituent systems during SoS operation.

3.2. Methodology of case study

A case study is conducted to demonstrate and evaluate the proposed CT concept for SoS development through quantitative and qualitative analysis. Such analysis evaluates the viability of the CT concept by comparing different scenarios for decision-making during SoS operations. The purpose of the case study is to construct cognitive twin ontology and support SoS decision-making through reasoning, so as to prove that the proposed cognitive twin is workable and meaningful.

- ***Case study definition:*** A case study of an SoS including three unmanned aerial vehicles (UAVs) and three unmanned surface vehicles (USVs) is conducted. SoSs are also constructed for cases containing more UAVs and USVs.[1]
- ***Scenario design:*** Each UAV can land on only one USV. The scenario is to determine an optimized solution for the UAV landing with the least time.
- ***Integrated simulation of the scenario:*** DTs are developed for each UAV and USV. HLA is used to support integrated simulations of the SoS by implementing communications across DTs.
- ***Reasoning for decision-making:*** Results from HLA simulations are formalized in ontology models. Through reasoning of the ontology models, the optimized solution for UAV landing is obtained for the real SoS operations.
- ***Evaluation:*** Qualitative and quantitative approaches are used to evaluate the expected purposes of the proposed CT according to their respective metrics:

[1] Available at https://gitee.com/zkhoneycomb/open-share/tree/lihan/Papers.

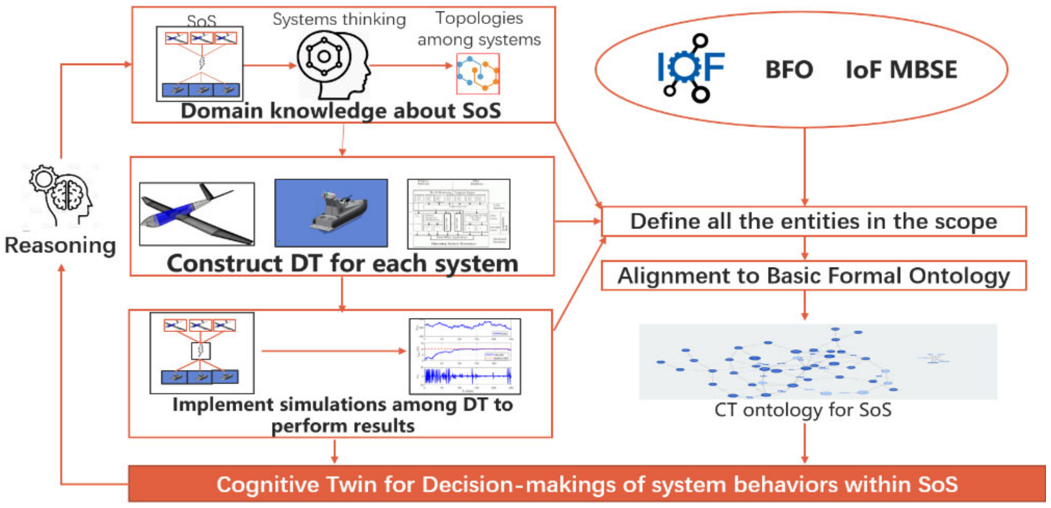
SoS
Systems thinking
Topologies among systems
Domain knowledge about SoS
IOF
BFO
IoF MBSE
Reasoning
Construct DT for each system
Define all the entities in the scope
Alignment to Basic Formal Ontology
Implement simulations among DT to perform results
CT ontology for SoS
Cognitive Twin for Decision-makings of system behaviors within SoS

Fig. 1. Construction of cognitive twins.

a) When using a qualitative approach, three metrics are proposed: 1) complete description, i.e., whether the designed ontology describes the scenario, HLA models, and DTs; 2) integrated simulation, i.e., whether the HLA enables the integration of different DTs of UAVs and USVs to predict the performance of the SoS; and 3) decision-making for SoS operations, i.e., whether the reasoning of ontology models enables decision-making for system behaviors within the SoS.
b) When using a quantitative approach, three metrics are considered: 1) the individuals (which refer to the instances in the ontology) developed based on the ontology in the case study; 2) the HLA entities generated from ontology models; and 3) the entire landing time of the optimized landing scenario determined by reasoning.

4. Cognitive twins for system of systems development

4.1. Overview

Figure 2 presents an overview of the proposed CT. The CT includes three main components:

– Virtual entities, which refer to the digital shadows, digital models, or digital presentations of the constituent systems in the SoS and the topology among them. The digital shadows contain the architecture model, simulation model, etc, as well as data information such as simulation results. Virtual entities are used to perform the system behaviors for each constituent system.
– Ontology, which refers to the concept definitions and topology between them, is used to support semantic modeling for formalizing the virtual entities, SoS, results of integrated simulation across DTs, and the topology among them. The ontology is developed based on the upper-level ontology BFO, IOF specifications, and the IOF model-based systems engineering ontology approach, GOPPRRE. Based on the reference ontologies, semantic modeling is used to develop ontology models for constructing the CT.
– SoS, which refers to the physical pairs of virtual entities. For example, in the case study, the SoS consists of three UAVs and three USVs.

SoS that is not the focus of research has been designed and manufactured. All physical entities and topologies have been defined. Virtual entities are composed of models constructed during SoS development, and the interfaces of each component system are implemented by HLA. Finally, ontology is constructed to formalize both SoS and virtual entitis.

Apart from the main CT components, HLA is used to support integrated simulation across DTs of constituent systems. To realize the automated simulation, digital twin federates (DTFs) for each of the digital models, or virtual entities, are generated from ontology models. Then, through HLA objects, such DTFs form the basis to construct an HLA federation model.

The federation model is used to implement HLA simulations through a run-time infrastructure (RTI) that

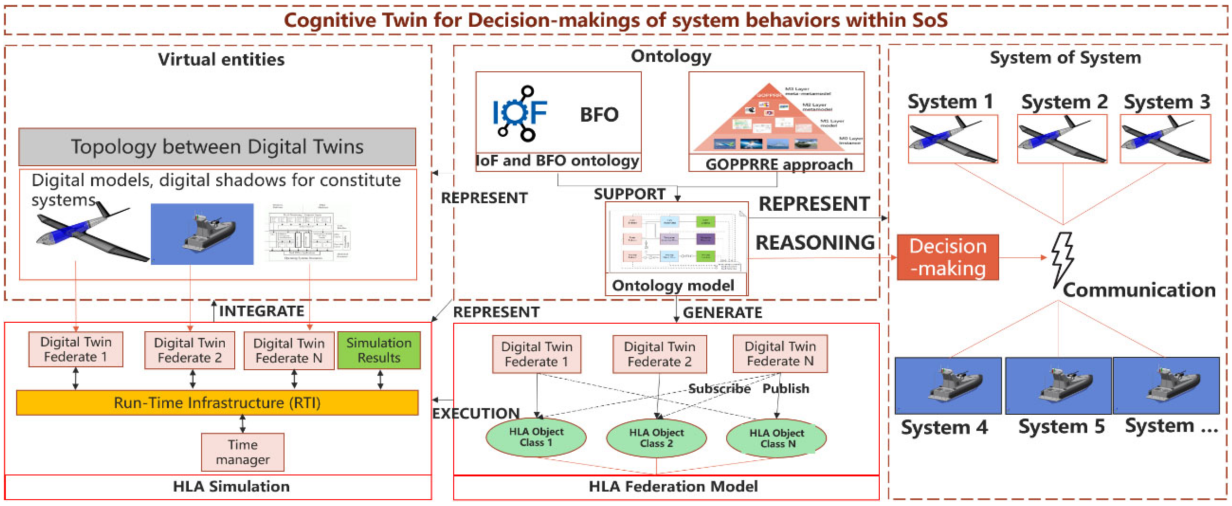

Fig. 2. The composition of the cognitive twins.

controls different DTFs to exchange their respective simulation data through a time manager. After simulation executions, the simulation results are generated and defined in the ontology models. Finally, through reasoning of the ontology models, decision-making is performed to control each constituent system to achieve better SoS performance.

4.2. *Ontology design for DT integration*

As shown in Fig. 3, the ontology for the SoS and DTs is defined based on the BFO and IOF specifications. The red nodes represent IOF and BFO ontology concepts. The purple nodes represent the domain ontology for the SoS and DTs. The blue nodes represent the scenario ontology. The *SoS and DT entities* class includes two subclasses: 1) *Occurrent*, referring to "an entity that unfolds itself in time or it is the instantaneous boundary of such an entity or it is a temporal or spatiotemporal region that such an entity occupies"; 2) *Continuant*, referring to "an entity that persists, endures, or continues to exist through time while maintaining its identity." The details of other classes are provided in [36]. Through the ontology framework, the specific SoS, its constituent systems, and the operational processes, functions, and qualities in the real world are formalized with the information of the DTs, HLA simulation model, and simulation results. Based on the ontology, the Web Ontology Language (OWL) is used to design the complete ontology for the SoS and DTs under the IOF and BFO ontology framework.

Definition 1. Ontology for SoS and DT

Token:: = refers to a collection of ontology concepts. Token $\sum$ refers to a collection of some given elements. *Ontology* refers to the ontology hierarchy we designed.

$$\begin{aligned} &Ontology:: = \{SoS ::= \{Coordinate, Artifact,\\ &Person, ArtiAggre, Organization), \}\\ &VEs:: = \{SimuResults, DT, HLAModels)\}\} \qquad (1)\end{aligned}$$

Where *SoS* refers to the physical entities in the real world. It contains *Coordinate*, *Artifact*, *Person*, *ArtiAggre*, and *Organization*. *Coordinate* refers to the coordinate system in physical space; *Artifact* refers to a physical element in the SoS; *Person* refers to relevant people in the SoS; *ArtiAggre* and *Organization* refer to the effective sets of artifacts and persons in the SoS. *VEs* refer to the virtual entities of the SoS. It contains *SimuResults*, *DT*, and *HLAModel*. *SimuResults* refers to the results of one distributed simulation. *DT* refers to a digital model of the constituent system in the SoS. *HLAModels* refers to the models that describe the topology among different DTs.

In this study, we employ the IOF MBSE ontology to represent *HLAModels* for implementing simulations automatically [34,35], as shown in Fig. 4. The ontology is developed based on an M0-M3 modeling framework: 1) The M0 layer represents the physical entities of DTs in the real world; 2) The M1 layer represents the virtual entities of DTs; 3) The M2 layer represents meta-models; 4) The M3 layer represents meta-meta smodels, including the elements: Graph, Object, Relationship, Property, Role, Point and Extention between them. The M0-M3 modeling framework provides standardized and semantic expressions for different model

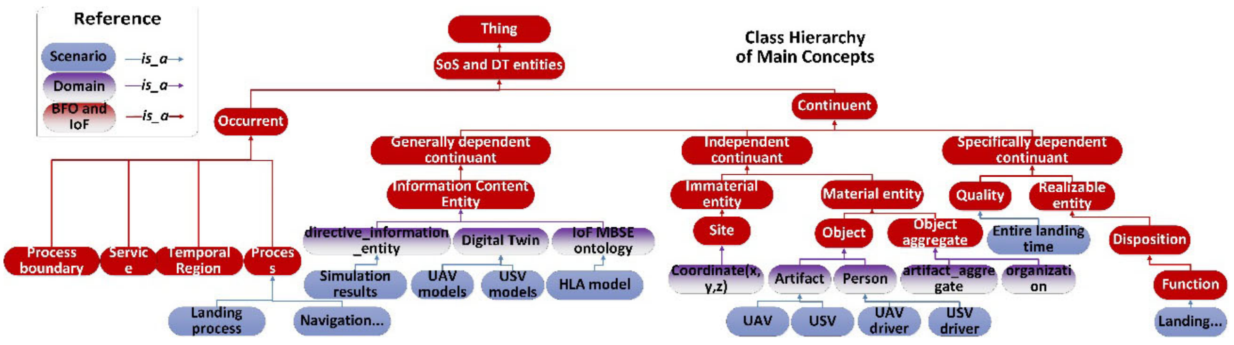

Fig. 3. Ontology class hierarchy based on BFO and IoF.

structures, which are used as the basis of the MBSE ontology definition.

OWL is applied to design an MBSE ontology based on GOPPRRE that can support information exchange across the DTs. To express the concepts clearly, some definitions are provided to describe the ontology concepts for the DTs and HLA models using the GOPPRRE approach [34,35].

Definition 2. The ontology of HLA models using OWL:

$$\mathit{HLAModels} = \{C, I_C, \mathit{OP}(i_A, i_B), \mathit{DP}(i_A, d)\} \quad (2)$$

The definition of *HLAModels* includes four components: C denotes an ontological concept set of the specific domain classes. I_C refers to a concept set of individuals, which are the instances of C. $\mathit{OP}(i_A, i_B)$ defines a set of object properties, which represent the relationships between two individuals i_A and i_B. $\mathit{DP}(i_A, d)$ denotes a set of datatype properties, which represent the attribute values of individuals.

Definition 3. Several ontology classes are introduced, where the subscript C refers to a root class that has subclasses:

$$C ::= \left\{\mathit{Graph}_{C_G}, \mathit{Object}_{C_O}, \mathit{Point}_{C_P}, \mathit{Property}_{C_{PR}}, \mathit{Relationship}_{C_{RE}}, \mathit{Role}_{C_R}\right\} \quad (3)$$

Based on the GOPPRRE approach, there are six root classes to represent the HLA model structure. The class Graph_{C_G} refers to the meta-meta model *Graph*. The classes Object_{C_O}, Point_{C_P}, $\mathit{Property}_{C_{PR}}$, $\mathit{Relationship}_{C_{RE}}$, and Role_{C_R} refer to the other five meta-meta models of the GOPPRR. The meta-models, which are used for constructing the HLA models, are subclasses of those root classes. The details of the HLA meta-models are introduced in Section 4.3.

Definition 4. Object properties are used to define the interrelationships among ontological individuals.

$$\mathit{OP}(i_A, i_B) ::= \{\mathit{Includes}(i_A, i_B), \mathit{HasPro}(i_A, i_B), \mathit{HasPo}(i_A, i_B), \mathit{LinkRo}(i_A, i_B), \mathit{Binding}(i_A, i_B)\} \quad (4)$$

The object property $\mathit{OP}(i_A, i_B)$ indicates the interrelationships between the individuals i_A and i_B. Based on the GOPPRRE approach, there are five different object properties that describe the interrelationships between individuals of the six main classes. The other concepts in Eq. (4) are introduced as follows:

Token $\equiv$ refers to the domains and ranges of $\mathit{OP}(i_A, i_B)$ (Definition 4) and $\mathit{DP}(i_A, d)$ (Definition 5). The left of token $\vdash$ represents the domains of object property and datatype property and the right represents their ranges.

$$\mathit{Includes}(i_A, i_B) \equiv \{\mathit{graph}_{\mathit{GTp}} \vdash (\mathit{object}_{\mathit{ObTp}}, \mathit{relationship}_{\mathit{reTp}})\} \quad (5)$$

The object property $\mathit{Includes}(i_A, i_B)$ indicates that a graph includes an object and relationship, where $\mathit{graph}_{\mathit{gTp}}$ is an individual of class *GTp*, which is a subclass of Graph_{C_G}. The $\mathit{object}_{\mathit{ObTp}}$ is an individual of class *ObTp*, which is a subclass of Object_{C_O}. The $\mathit{relationship}_{\mathit{reTp}}$ is an individual of class *ReTp*, which is a subclass of $\mathit{Relationship}_{C_{RE}}$.

$$\mathit{HasPro}(i_A, i_B) \equiv \{\mathit{nonpro}_{\mathit{NonproTp}} \vdash \mathit{property}_{\mathit{ProTp}}\} \quad (6)$$

The object property $\mathit{HasPro}(i_A, i_B)$ indicates that an individual (model instance) has certain properties, where $\mathit{nonpro}_{\mathit{NonproTp}}$ refers to the individual of class *NonproTp*, which is a subclass of Graph_{C_G}, Object_{C_O}, Point_{C_P}, $\mathit{Relationship}_{C_{RE}}$, or Role_{C_R}. $\mathit{property}_{\mathit{ProTp}}$ refers to the individual of $\mathit{Property}_{C_{PR}}$'s subclass.

$$\mathit{HasPo}(i_A, i_B) \equiv \{\mathit{object}_{\mathit{ObTp}} \vdash \mathit{point}_{\mathit{PoTp}}\} \quad (7)$$

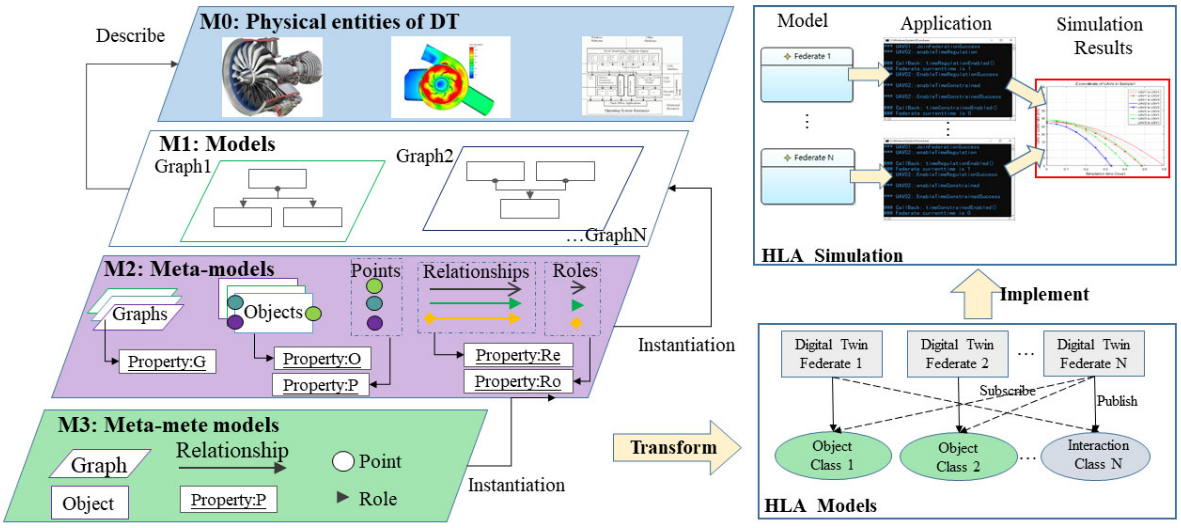

Fig. 4. GOPPRRE Formalizing HLA models.

The object property $HasPo(i_A, i_B)$ indicates that an object has one point, where $point_{PoTp}$ refers to the individual of class *PoTp*, which is a subclass of $Point_{C_P}$.

$$LinkRo(i_A, i_B) \equiv \{relationship_{ReTp} \vdash role_{RoTp}\} \quad (8)$$

The object property $LinkRo(i_A, i_B)$ indicates that a relationship instance links a role on one side, where $role_{RoTp}$ refers to the individual of class *RoTp*, which is a subclass of $Role_{C_R}$.

$$Binding(i_A, i_B) \equiv \{role_{RoTp} \vdash (object_{ObTp}, point_{PoTp})\} \quad (9)$$

The object property $Binding(i_A, i_B)$ indicates how an object (or a point) binds with a relationship. The individual $role_{RoTp}$ can bind with an $object_{ObTp}$ or its point $point_{PoTp}$.

Definition 5. Datatype properties are used to define the interrelationships between individuals and values.

$$DP(i_A, d) ::= \{Value(i_A, d)\} \quad (10)$$

Datatype property $DP(i_A, d)$ of the ontology indicates the interrelationships between the individual i_A and value whose datatype is d. Based on the GOPPRRE approach, they are unified into one datatype property.

$$Value(i_A, d) \equiv \{property_{ProTp} \vdash \varnothing\} \quad (11)$$

The GOPPRRE approach uses the meta-meta model Property to describe the attribute of the other five meta-meta models. Therefore, there is only one necessary datatype property $Value(i_A, d)$, where i_A is an individual $property_{ProTp}$ of class *proTp*, which is a subclass of $Property_{C_{PR}}$, and $\varnothing$ indicates that the value of $property_{ProTp}$ can be any datatype, such as *int*, *long*, or *byte*.

Definition 6. Connection is a directed link between Object individuals.

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

$$connector ::= \{LinkRo(relationship_{ReTp}, role_{RoTp}), Binding(role_{RoTp}, point_{RoTp}), HasPo(object_{ObTp}, point_{PoTp})\} \quad (12)$$

Where the *connector* defines a binding between one *Point* or *Object* and one *Role* on one side of the *Relationship*. It contains three object properties that allow the individual $relationship_{ReTp}$, $role_{RoTp}$, and $object_{ObTp}$ (or $point_{PoTp}$ in $object_{ObTp}$) to be connected.

With the definition of *connector*, the concept of a *connection* is defined as a directed link between two different *Object* individuals in one model, which is realized by a *Relationship* individual. Token is defined as a *connection* that is linked from a to b, created based on two *connector* constraints. The concept of a *connection* is formally defined as follows:

$$connection ::= connector \Rightarrow connector'$$
$$= \{LinkRo(relationship_{ReTp}, role_{RoTp}),$$
$$Binding(role_{RoTp}, point_{PoTp}),$$
$$HasPo(object_{ObTp}, point_{PoTp})\} \Rightarrow$$

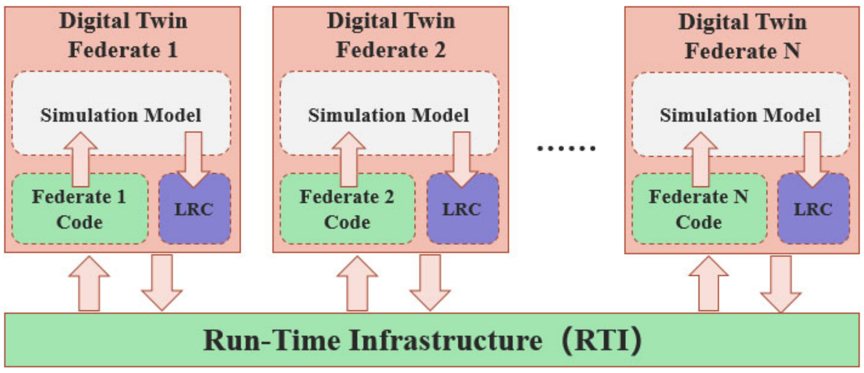

Fig. 5. A digital twin federation based on HLA.

$$\begin{aligned}&\{LinkRo(relationship_{ReTp}, role'_{RoTp}),\\&Binding(role'_{RoTp}, point'_{PoTp}),\\&HasPo(object'_{ObTp}, point'_{PoTp})\}\end{aligned} \tag{13}$$

Where the *connection* is defined based on *connector* and *connector'*. Through this definition, one individual $object_{ObTp}$ and another $object'_{ObTp}$ are linked by one *Relationship* individual $relationship_{ReTp}$, and the directed links representing the logic in the MBSE models are demonstrated.

4.3. Integrated simulation across DTs based on HLA

To define standardized interfaces of heterogeneous DTs, a distributed simulation based on HLA is adopted, which is a universal framework for distributed simulation standardized by the IEEE. HLA provides a set of rules based on the following main concepts to achieve interoperability and reusability when constructing the standardized interfaces for DTs:

– Federation: A simulation application composed of a set of simulation components.
– Federate: One simulation component that represents the basic elements of HLA.
– RTI: Simulation-oriented middleware for managing alliance interaction.

These specifications effectively define interfaces across DTs. In HLA, a federate is the basic component for the distributed simulation. When integrating DTs and HLA, the federate and virtual entities construct a DTF. Thus, each DT has its respective DTF referring to a standardized interface to the RTI. To generate compiled code for executing co-simulations, semantic models express the interactions between DTFs.

Figure 5 shows a digital twin federation based on HLA. The federation contains a number of DTFs based on the HLA interface specification and OMT. A DTF is composed of simulation models referring to virtual entities of the DT, federate code, and local RTI component (LRC). The simulation model consists of federate objects that meet the HLA standard aiming to construct the data exchange interface of each virtual entity for the request and response to the RTI using the HLA interaction specification. The federate code is an application program that performs local DT executions, which defines interactions with the RTI. The LRC is the API component for the DTF to communicate with the RTI.

The MBSE ontology in Section 4.2 is used to describe the data exchange between different DTFs and develop the HLA models. The HLA model is defined by the *federation object model (FOM)*, which describes the topology among HLA federates during the federation execution. The *FOM* includes *object classes* and *interaction classes*. An *object class* defines a physical entity of an SoS, and it is composed of a set of *object attributes* whose values define the state of the entity. An *interaction class* defines an event that occurred during the *federation* execution, and *interaction parameters* define the information used in the event. The *publish* and *subscribe* mechanisms are used to support the information exchange during *federation* execution. When one *federate publishes* an *object class*, it registers an instance of an *object class*, and updates its attribute values. Other *federates* that *subscribe* to the same *object class* can obtain the related value instances and then receive the attribute value. The instances of *interaction classes* are used similarly, except that they do not represent persistent entities.

Definition 7. Based on the ontological concepts proposed in Section 4.2, the HLA models are formalized using several subclasses of GOPPRRE:

$$Graph_{C_G} \ni \{Graph_{C_G}^{Federation}\} \tag{14}$$

Token $\ni$ refers to the collection of subclasses. The class $Graph_{C_G}^{Federation}$, which is a subclass of $Graph_{C_G}$, refers to a meta-model of Graph. One individual of $Graph_{C_G}^{Federation}$ represents an HLA federation in this ontology, which is the basic element of an integrated simulation.

$$Object_{C_O} \ni \{Object_{C_O}^{Fed}, Object_{C_O}^{Spa}, Object_{C_O}^{Obj}, Object_{C_O}^{Int}, Object_{C_O}^{Att}, Object_{C_O}^{Par}\} \tag{15}$$

$Object_{C_O}^{Fed}$ refers to the meta-model of Object whose type is federate *Fed*, which indicates a simulation component that represents the basic elements of HLA. $Object_{C_O}^{Spa}$ refers to the meta-model of Object whose type is path space *Spa*, which represents the virtual space in the HLA simulation. $Object_{C_O}^{Obj}$ refers to the meta-model of Object whose type is object class *Obj*, which represents a virtual entity in the federate, such as a UAV model. $Object_{C_O}^{Int}$ refers to the meta-model of Object whose type is interaction class *Int*, which represents the information exchanged between federations. $Object_{C_O}^{Att}$ refers to the meta-model of Object whose type is object attribute *Att*, which represents the state of a persistent object, such as the location of a UAV model. $Object_{C_O}^{Par}$ refers to the meta-model of Object whose type is interaction parameter *Par*, which represents the property of an event that occurred during the federation execution.

$$Point_{C_P} \ni \{Point_{C_P}^{(Obj,In)}, Point_{C_P}^{(Obj,Out)}, Point_{C_P}^{(Int,In)}, Point_{C_P}^{(Int,Out)}\} \tag{16}$$

$Point_{C_P}^{(Obj,In)}$ and $Point_{C_P}^{(Obj,Out)}$ refer to the meta-models of Point that are in $Object_{C_O}^{Obj}$. $Point_{C_P}^{(Int,In)}$ and $Point_{C_P}^{(Int,Out)}$ refer to the meta-models of Point that are in $Object_{C_O}^{Int}$. *In* represents that this point is the input port for information exchange between federates, and *Out* is the output port.

$$Relationship_{C_{RE}} \ni \{Relationship_{C_{RE}}^{Pub}, Relationship_{C_{RE}}^{Sub}, Relationship_{C_{RE}}^{Has}\} \tag{17}$$

$Relationship_{C_{RE}}^{Pub}$ refers to the meta-model of Relationship whose type is publish *Pub*; it connects $Object_{C_O}^{Fed}$ and $Object_{C_O}^{Obj}$ or $Object_{C_O}^{Int}$, which indicates that the federate is publishing an *object* or *interaction class*. $Relationship_{C_{RE}}^{Sub}$ is used in a similar manner, except that it refers to subscribing. $Relationship_{C_{RE}}^{Has}$ refers to the meta-model of Relationship whose type is *Has*; it connects $Object_{C_O}^{Obj}$ and $Object_{C_O}^{Att}$ or $Object_{C_O}^{Int}$ and $Object_{C_O}^{Att}$, which indicates that the *object* or *interaction class* has an *attribute* or *parameter*.

$$Role_{C_R} \ni \{Role_{C_R}^{(Pub,From)}, Role_{C_R}^{(Pub,To)}, Role_{C_R}^{(Sub,From)}, Role_{C_R}^{(Sub,To)}, Role_{C_R}^{(Has,From)}, Role_{C_R}^{(Has,To)}\} \tag{18}$$

$Role_{C_R}^{(Pub,From)}$ and $Role_{C_R}^{(Pub,To)}$ refer to the meta-models of Role belonging to $Relationship_{C_{RE}}^{Pub}$. $Role_{C_R}^{(Pub,From)}$ is used to connect a federate, and $Role_{C_R}^{(Pub,To)}$ is used for an *object* or *interaction class*. $Role_{C_R}^{(Sub,From)}$ and $Role_{C_R}^{(Sub,To)}$ belong to $Relationship_{C_{RE}}^{Sub}$ and are used in a similar manner. $Role_{C_R}^{(Has,From)}$ and $Role_{C_R}^{(Has,To)}$ refer to the meta-models of Role belonging to $Relationship_{C_{RE}}^{Has}$. $Role_{C_R}^{(Has,From)}$ is used to connect an *object* or *interaction class*, and $Role_{C_R}^{(Has,To)}$ is used for an *attribute* or *parameter*.

$$Property_{C_{PR}} \ni \{Property_{C_{PR}}^{(Obj,PS)}, Property_{C_{PR}}^{(Int,PS)}, Property_{C_{PR}}^{(Att,TM)}, Property_{C_{PR}}^{(Int,TM)}\} \tag{19}$$

$Property_{C_{PR}}^{(Obj,PS)}$ and $Property_{C_{PR}}^{(Int,PS)}$ refer to the meta-models of Property belonging to $Object_{C_O}^{Obj}$ and $Object_{C_O}^{Int}$; this type is *PS*, which means this object class could be published or subscribed by federates. $Property_{C_{PR}}^{(Att,TM)}$ and $Property_{C_{PR}}^{(Int,TM)}$ refer to the meta-model of Property belonging to $Object_{C_O}^{Att}$ and $Object_{C_O}^{Int}$; this type is *TM*, which indicates a time management strategy.

In summary, the integration of DTs within an SoS is supported by HLA using its standardized interface specification. The communication between heterogeneous DTs can be realized through the interfaces provided by the RTI. In addition, through the ontology based on the GOPPRRE approach mentioned in Section 4.2, the HLA models are formalized by the ontology model in CT.

4.4. Generating and executing simulation models from semantic models

The MBSE ontology of CTs that describes the topology among different DTs provides a unified description of the entire physical SoS and virtual models. However, heterogeneous interfaces and interface management when integrating DTs is supported by HLA. Thus, an integrated simulation framework is developed for generating HLA execution models using an automated transformation algorithm.

To realize this transformation, the ontology of CTs is used to generate *federations* through a code transformer: 1) Apache Jena is used to load an ontology model representing HLA *federations* and the topology among them; 2) A generator is developed to traverse the entire loaded model and generate the C++ code for HLA simulation executions. The overall workflow of the transformation is defined as shown in Algorithm 1. Algorithm 1 takes as input the MBSE ontology *HLAModels* and provides as output the *Federation* that implements the HLA simulation. All individuals of *Graph* representing one HLA *federation* are queried to execute separately to generate the *federation* file: 1) the individuals of each $Object_{CO}^{Obj}$ included by $g(i)$ representing *federate* are loaded to create *DTF* in the given *federation*; 2) the individuals of $Relationship_{C_{RE}}^{Pub,Sub}$ represent that the given *DTF* is *publishing* or *subscribing* to some *object classes* or *interaction classes*, such that all individuals of $Relationship_{C_{RE}}^{Pub,Sub}$ are traversed to create *object and interaction classes*, which are *published* or *subscribed* by *DTF*. The generation of *federation* is completed after all *DTFs* and all *object* and *interaction classes* are created.

Algorithm 1: Transformation from ontology to HLAsimulation system

```
Input: HLAModels
Output: $\sum federation$
for Individual $g(i)$ from $Graph_{CG}^{Federation}$ do
    Create federation(i)
    for Individual ob(j) from $Object_{CO}^{Obj}$ do
        if Includes(g(i), ob(j)) then
            Create DTF(j) which joined in federation(i).
            for Individual re(k) from $Relationship_{C_{RE}}^{Pub,Sub}$ do
                if LinkRo(re(k), ro(k)) and Binding(ro(k), po(k)) and
                HasPo(ob(k), po(k)) then
                    Create Obj/IntClass(k) which published/subscribed
                    by DTF(j).
                end if
                k++
            end for
        end if
        j++
    end for
    i++
end for
return $\sum federation$
```

After the *federations* are generated, the HLA simulation is executed through the RTI, which is a time manager application developed in accordance with the HLA interface specification. The RTI provides simulation operational services for simulation execution, such as simulation start, pause, resume, and time synchronization. Moreover, the RTI provides the communication transmission services, which implement each *federate* independently.

Algorithm 2: The process of HLA federation execution

```
Input: $\sum federation$, et the end time of simulation
Output: $\sum$ Instances of ObjClass and IntClass
for federation from $\sum federation$ do
// Create federation execution
    if federation not in run then
        DTF.createFederationExecution();
    end if
        DTF.joinFederation();
// Start federation execution
    if DTF publish ObjClass then
        $ObjClass\ ob = DTF$.registerInstance();
    end if
    if DTF subscribe ObjClass' then
        $ObjClass'\ ob' = DTF$.registerInstance();
    end if
// Execute federation
    Set the current time of simulation $ct = 0$
    while ct < et do
        DTF.updateAttribute(ob);
        DTF.reflectAttribute(ob');
        if DTF publish IntClass then
            $IntClass\ in = DTF$ sendInteraction();
        end if
        if DTF subscribe IntClass' then
            $IntClass'\ in' = DTF$ receiveInteraction ();
        end if
            DTF.timeAdvance(ct);
            DTF.synchronous();
        end while
// Destroy federation execution
        DTF.resignFederationExecution();
        DTF.destroyFederationExecution();
end for
return $\sum$ Instances
```

The process of *federation* execution is shown in Algorithm 2. Each DTF in this *federation* is an independent simulation agent and executed simultaneously, and all services used by the DTF are provided by the RTI. The input of *federation* execution is the *federation* file, which is generated by Algorithm 1 with the related simulation parameters. Then, some instances of object classes or interaction classes are created during execution. The execution process involves four activities:

Create federation execution: *Federation* execution, which refers to an integrated simulation process, is created by one DTF. Then, all the other DTFs join in.

Start federation execution: DTF publishes and subscribes to *object classes* and *interaction classes* during this process. Then, instances of *object classes* are created and they exist throughout the *federation* execution.

Execute federation: In this process, there are two processes of data exchange that are controlled by instances of *object classes* or *interaction classes*.

Destroy federation execution: After all data exchange is completed, *federation* execution is destroyed by DTF and the simulation results are obtained.

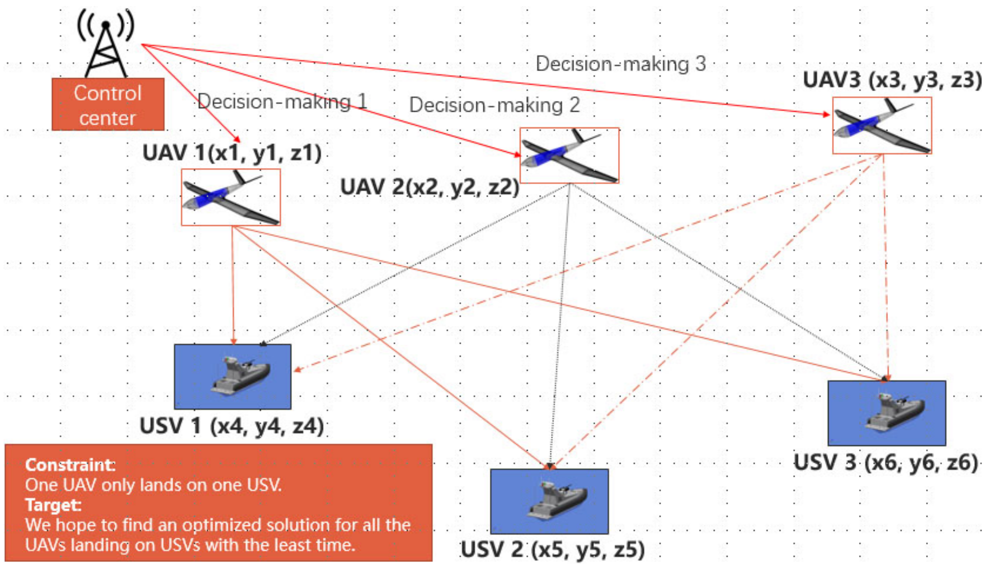

Fig. 6. Problem statement of UAV landing.

5. Case study

5.1. Problem statement

As shown in Fig. 6, an SoS consisting of three UAVs, three USVs, and one control center is demonstrated. The UAVs are planning to land on the USVs. Each UAV lands on only one USV. Thus, each UAV has three options for landing. The control center provides decision-making for each UAV regarding which USV they land on. All the USVs and UAVs have their own locations (x, y, z) in the fixed space coordinate system where z represents height, and the USVs are stationary. Whenever a UAV lands, the control center uses its current location to support decision-making for the other UAVs. The target of the control center is to complete the entire landing process in the least time.

All systems in the system have been designed and there are physical pairs of virtual entities. To develop a CT for this target, ontology models for this scenario are first constructed, including a scenario description, HLA model description, and simulation result descriptions (which are created after the simulations). Moreover, the DTs for each UAV and USV are developed separately. One HLA simulation model is developed to support communications among different DTs for implementing an integrated simulation of the landing time. As shown in Table 1, ten datasets are provided to define the initial locations of UAVs, and the fixed locations of USV1, USV2 and USV3 are (0,0,0), (10,10,0) and (20,0,0). Then, HLA simulations are implemented for the six landing scenarios separately (for example, UAV1 landing on USV1, UAV2 landing on USV2, UAV3 landing on USV3) under each set of initial locations. The total time of the entire landing process is captured through the HLA simulation for each landing scenario. Finally, based on the ontology models and DT integration based on HLA, a CT is constructed for operational decision-making of each UAV by the control center. All the landing scenarios are shown in Table 2.

Table 1
Sets of initial locations of UAVs (km)

	UAV1	UAV2	UAV3
Sample1	(24,47,28)	(26,18,27)	(12,39,29)
Sample2	(36,38,40)	(45,10,22)	(40,2,36)
Sample3	(30,25,33)	(20,45,21)	(36,14,28)
Sample4	(16,0,21)	(41,48,35)	(14,32,20)
Sample5	(5,18,13)	(42,3,28)	(2,18,22)
Sample6	(23,10,11)	(13,10,24)	(37,4,38)
Sample7	(44,6,30)	(22,3,28)	(21,26,23)
Sample8	(34,4,25)	(43,21,31)	(27,7,32)
Sample9	(18,36,17)	(2,42,40)	(14,22,21)
Sample10	(1,20,36)	(40,32,22)	(17,30,38)

5.2. Semantic modeling

To support decision-making of the proposed SoS in the case study, ontology models are built to formalize the SoS using the ontology proposed in Sections 4.2 and 4.3. Then, HLA simulation models are generated from the ontology models as in Section 4.4. After the simulations are executed by the HLA models, the simulation results are synchronized into the ontology mod-

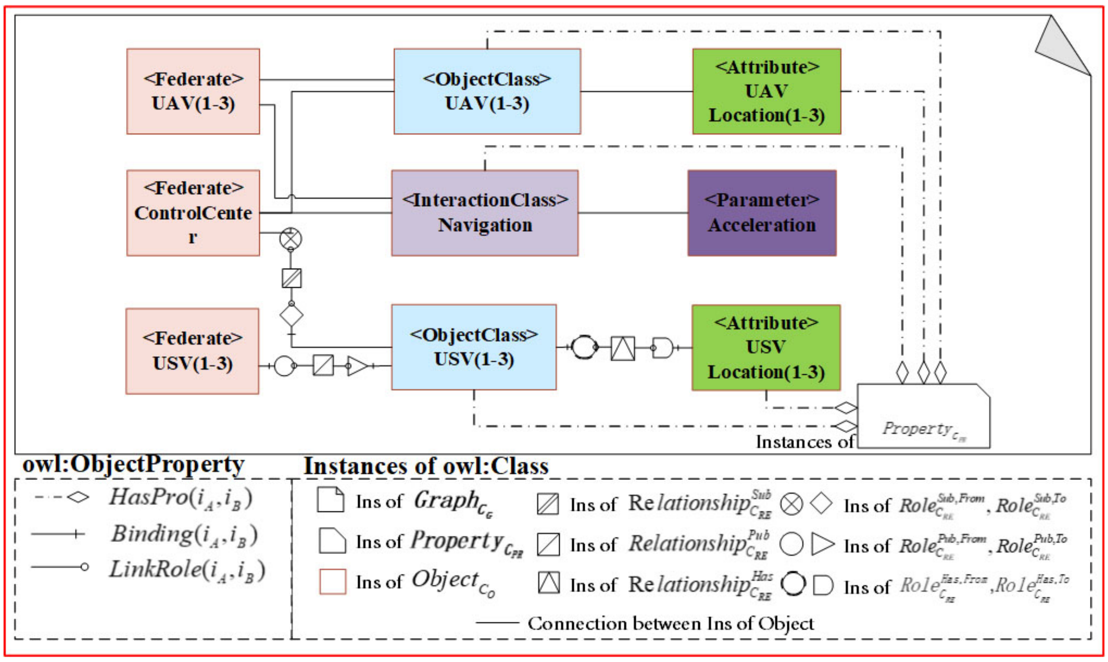

Fig. 7. The ontology of UAV landing.

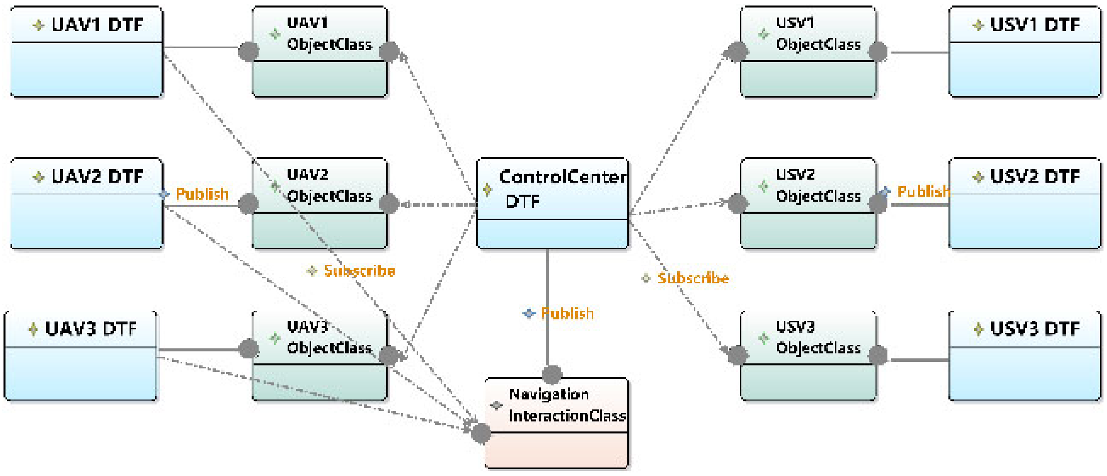

Fig. 8. HLA model of UAV landing.

els. Finally, the ontology models are used to support decision-making for the SoS operations by reasoning.

As shown in Eq. (1), the ontology models include the topology among DTFs in different HLA models, their simulation results, and the real SoS scenarios. The SoS in the case consists of systems including UAV, USV and control center, and includes relevant people. According to BFO and IOF specifications, we have the following definition:

$$Artifact \ni \{UAV, USV, ControlCenter\} \tag{20}$$

$$Person \ni \{UAVDriver, USVDriver\} \tag{21}$$

Where *UAV*, *USV* and *ControlCenter* are subclasses of *Artifact UAVDriver* and *USVDriver* are subclasses of *Person*. UAVs, USVs and the control center are individuals of thoes class.

The ontology model for the topology includes instances of Graph, Object, Property, Point, Role, and Relationship according to the landing scenario. Figure 7 is a individual of $Graph_{CG}^{Federation}$, which represents a topology among DTs of UAVs and USVs. It contains some

Table 2
Six landing scenarios of UAVs and USVs

Scenario1	UAV1 landing on USV1, UAV2 landing on USV2, UAV3 landing on USV3
Scenario2	UAV1 landing on USV1, UAV2 landing on USV3, UAV3 landing on USV2
Scenario3	UAV1 landing on USV2, UAV2 landing on USV1, UAV3 landing on USV3
Scenario4	UAV1 landing on USV2, UAV2 landing on USV3, UAV3 landing on USV1
Scenario5	UAV1 landing on USV3, UAV2 landing on USV2, UAV3 landing on USV1
Scenario6	UAV1 landing on USV3, UAV2 landing on USV1, UAV3 landing on USV2

individuals of classes which defined in Eqs (15–19). The line in this graph represents connection which is defined as a directed link between two different Object individuals in one model. The ontology model describes the attributes of the DTs of each UAV and USV and all the interactions between them.

There are seven DTFs in this case, three *UAV DTFs*, three *USV DTFs*, and one *ControlCenter DTF*. *UAV DTFs* publish *UAV Object Class* and *USV DTFs* publish *USV Object Class*, which are both subscribed to by *ControlCenter DTF*. When the instances of *UAV Object Class* and *USV Object Class*, which represent the real entities in the SoS, are moving, *ControlCenter DTF* receives their coordinate changes. *Navigation Interaction Class* is published by *ControlCenter DTF* and subscribed to by *UAV DTFs*. *ControlCenter DTF* calculates the acceleration of the UAVs according to the coordinates and the landing scenario, and then sends the acceleration information to the UAVs through the instances of *Navigation Interaction Class*. Finally, the UAVs land on the USVs according to the expected landing scenario.

5.3. *Integrated simulation based on HLA*

As shown in Fig. 8, HLA models are generated from ontology models. The simulation is executed using the algorithms in Section 4.4. The ontology models for the six landing scenarios are used to generate the HLA models separately with 10 sets of initial location information, as shown in Table 1.

When executing the HLA simulation, CERTI (an open-source HLA RTI software) is used to support communications among DTs of UAVs and USVs through their respective DTF simulation applications. Such applications are managed by a run-time infrastructure gateway (RTIG), which is used to control the process of coordinating HLA federates. Each federation must have at least one RTIG process. Each federate simulation application is developed based on the ontology model and CERTI APIs.

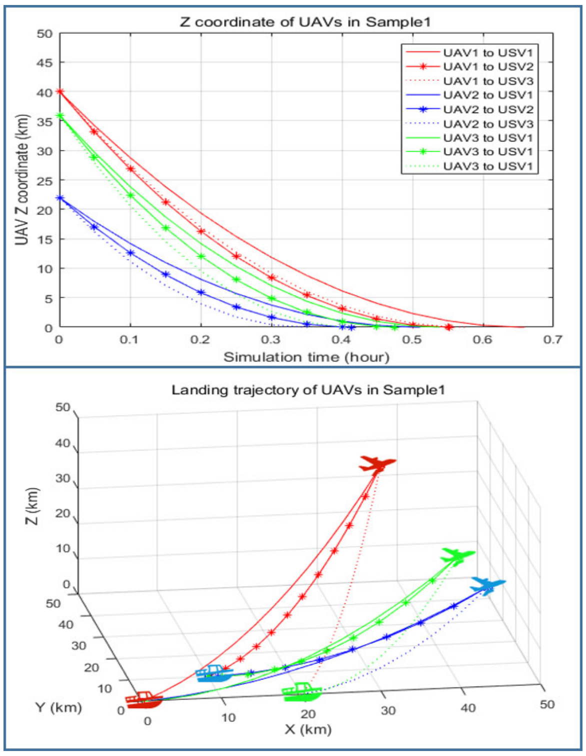

Fig. 9. Simulation results of UAV landing.

Sixty simulations were executed in total; the overall landing times for each landing scenario under all sets of initial locations are listed in Table 3, and one sample is shown in Fig. 9. From the results, we find that at least one landing scenario has the shortest overall landing time within each set of initial locations of HLA models. Thus, we infer that the landing scenario with the shortest overall landing time is the expected landing process when the UAVs and USVs are at the given initial locations. Based on the simulation results from HLA, the landing scenario and initial coordinates are formalized in ontology models through a developed synchronized Java plugin.

5.4. *Reasoning supporting decision-making*

Through DT integration based on HLA and semantic modeling for the topology of HLA models, simulation results, and the real SoS, the developed CT is used to support real-time decision-making when each constituent system operates in the SoS. In this case, when a set of real-time positions is given, the reasoning of the ontology models for each initial location is used

Table 3
Simulation results of the overall landing time (h)

	Scenario1	Scenario2	Scenario3	Scenario4	Scenario5	Scenario6
Sample1	0.5974	0.5974	0.4925	0.5006	0.5485	0.5485
Sample2	0.6588	0.6588	0.5532	0.5532	0.5745	0.5745
Sample3	0.5113	0.5113	0.5354	0.4966	0.5354	0.4771
Sample4	0.6025	0.6301	0.7218	0.6301	0.7218	0.6025
Sample5	0.4309	0.3574	0.5057	0.3574	0.5057	0.4309
Sample6	0.4182	0.4700	0.4182	0.5319	0.4700	0.5319
Sample7	0.5359	0.5359	0.4552	0.4552	0.3888	0.4057
Sample8	0.4659	0.4394	0.5702	0.4394	0.5702	0.4659
Sample9	0.5185	0.6073	0.5803	0.6073	0.5803	0.5185
Sample10	0.4851	0.4368	0.5575	0.5131	0.5575	0.5131

to obtain the optimal landing scenario for each UAV. The process consists of two parts: 1) the preprocessing of the real-time coordinates of each UAV and USV to confirm to which of the 10 sets of sample data (*Sample* in Table 1) the given real-time coordinate set belongs; 2) the reasoning algorithm is used to query the shortest landing time in the ontology model based on SPARQL queries.

Algorithm 3 is first used to compare the real-time positions of UAVs with the 10 sample datasets to determine which set to use for decision-making. The input elements of the algorithm are *RealTime*, which represents the set of real-time coordinates of UAVs, $\sum$ *Sample(i)*, which represents the 10 existing sample datasets, and *rg*, which refers to the range used to evaluate whether the real-time location can be replaced by sample data. All the sample data are traversed and compared with *RealTime*; if the difference between each coordinate of each UAV and *Sample(i)* is less than *rg*, then this *Sample(i)* is selected for decision-making.

After preprocessing the initial locations in Algorithm 3, the *SelectedSample* is provided to Algorithm 4. Algorithm 4 is developed to identify the landing scenario with the shortest landing time under the given initial locations through a SPARQL query of the developed ontology model. In the ontology model, the 10 sample datasets of initial locations and their related simulation results under each landing scenario are formalized. *bfo:SelectedSample* refers to the individual in the ontology that represents the *SelectedSample*. Through this query, all the six landing scenarios are sorted by their respective landing times, and the query result is the landing scenario with the shortest landing time. Through this query, the expected landing scenario is obtained from the CT. Based on the reasoning results, the control center in the SoS is able to make decisions regarding how each UAV lands on the determined USV.

Algorithm 3: Preprocessing of the given coordinate set

```
Input: RealTime = $\{UAV_1\ (x_1, y_1, z_1),\ UAV_2\ (x_2, y_2, z_2),$
    $UAV_3\ (x_3, y_3, z_3)\}$
    $\sum Sample(i) = \{UAV'_1\ (x'_1, y'_1, z'_1),$
    $UAV'_2\ (x'_2, y'_2, z'_2),\ UAV'_3\ (x'_3, y'_3, z'_3)\}$
    $rg = 3km$
Output: SelectedSample
//Loop all samples
for Sample(i) from $\sum$ Sample(i) do
    $j = 1$;
    $num = 0$;
/* Loop three UAVs and compare coordinates of RealTime and
Sample(i) */
    while $j <= 3$ do
    if $|x_j - x'_j| < rg$ and $|y_j - y'_j| < rg$ and $|z_j - z'_j| < rg$
    then
        num ++;
    end if
    j ++
    end while
/*If all coordinates meet the condition, then Sample(i) is selected*/
    if num == 3 then
        SelectedSample = Sample(i);
    end if
end for
return SelectedSample
```

In the Case Study, we used an SoS consisting of three UAVs, three USVs and a control center as a case. It is a complete SoS, which contains multiple systems, and each system has interaction. The UAVs, USVs and the control center are independent and heterogeneous systems, which makes complexity management and interface integration necessary. The systems need to communicate various information including speed, acceleration and attitude, and decisions need to be made as soon as possible for landing. It is necessary to construct CT ontology for this SoS and support decision-making through reasoning, so as to prove that the proposed CT is workable and meaningful.

6. Discussion and evaluation

In this section, quantitative and qualitative analyses of the CT are presented from three perspectives in

Algorithm 4: SPARQL algorithm for verifying the completeness of the MBSE models

```
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 bfo: <http://www.industrialontologies
.org/core/>
/*?scenario represents the optimal landing scenario and ?time
represents landing time*/
SELECT ?scenario, ?time
WHERE {
/*bfo:SelectedSample represents the selected sample data and it
is the input.*/
    bfo:SelectedSample bfo:hasProcess ?process.
    ?process bfo:hasResult ?result.
    ?result bfo:LandingTime ?time.
    ?process rdf:type ?scenario.
    ?scenario rdfs:subClassOf bfo:LandingScenario.
}
/*All landing scenarios are queried but only the one with the
shortest time is selected*/
order by ?time
limit 1
```

Table 4
OWL classes in the use case

Axiom	Sum	Real SoS	Virtual HLA model and DT
Class	52	7	45
Object property	18	6	12
Datatype property	11	9	2
Individual	153	23	130
Annotation property	67	4	27

the case study: 1) complexity management of CT for the SoS development; 2) DT integration for each constituent of the system model; 3) decision-making for real SoS operations based on CT.

6.1. Quantitative analysis

All the axioms in the entire ontology model for the use case are shown in Table 4. Among them, there are entities for the real SoS. The number of entities for real virtual models is 130. From the axiom of the ontology models, we infer that all the ontology concepts of the real SoS entity and virtual entity are developed. There are 153 individuals in the ontology models, and 46 connections among them. This is used to represent the topology among entities and manage the complexity of SoS development and operation.

In the case study, there are seven DTs including three UAVs, three USVs, and a control center in the SoS, as shown in Table 5. Moreover, all the corresponding DTs are integrated in one *federation* by their respective DTFs based on the HLA interface specification. This

Table 5
DT and HLA concepts

Use case concepts	Account
Initial locations	10
Landing scenario	6
DTF	7
HLA federation	1

Table 6
Individuals for reasoning and query result

Use case concepts	Account
Individuals before reasoning	120
Individuals after reasoning	2
Query time	0.15s

HLA *federation* is executed 60 times with 10 sample datasets and six landing scenarios. From Table 5, we find that all the DTs are integrated to predict the overall landing time of each landing scenario. Moreover, during the entire generation process of the HLA model, all the HLA DTFs and the *federation* are generated and implemented automatically.

In the case study, all 60 landing processes under different initial locations and different landing scenarios are executed, whose results are defined as individuals in the ontology model. As shown in Table 6, 60 individuals of the landing processes and 60 individuals of the simulation results are created. Those individuals are the cues for implementing a reasoning algorithm in Section 5.4, which is used to support real-time decision-making in the SoS. When a set of real-time positions of the UAVs and USVs is given, there are six landing scenarios as decision options. After the query, two individuals, of which one represents the optimal landing scenario and the other represents the shortest landing time, are given. The average time of this query is 0.15 s. Through this query, the optimal landing scenario is obtained through reasoning from 120 individuals.

6.2. Qualitative analysis

When constructing the CT, the CT ontology is definded by Eq. (1). All entities of SoS have been aligned to the BFO and IOF specification. Physical entities of UAV and USV are mapped to individuals of class *Artifact*; person of SoS is mapped to individuals of class *Person*; the simulation models and results are mapped to individulas of class *HLAModel* and *SimuResults.* Through this unified ontology, the topology between the virtual entities and real SoS is defined as a basis to implement reasoning to make decisions for the SoS operations. Moreover, the unified ontology provides a bridge between the real-world entities and their

DTs, because the related ontology models provide readable information for the computer to distinguish and describe the virtual DT and the real world. Furthermore, the unified ontology provides better scalability to extend the CT to other scenarios in the future: 1) the adopted BFO ontology is one of the most powerful upper-level ontologies, which has been used by the IOF (https://www.industrialontologies.org/); 2) the ontology for describing HLA models is defined based on the GOPPRRE approach, whose meta – meta models have excellent descriptive capabilities [27,34].

When constructing the CT, heterogeneous DTs are integrated through HLA specifications. When implementing HLA simulations, heterogeneous DTs are connected to the RTI by HLA *federates*. Through the RTI, run-time and synchronized communications can be managed and controlled among different DTs. Such HLA specifications are the basis to construct a unified platform to integrate different DTs of constituent systems in the SoS and implement simulations among them. Traditioal SoS development based on HLA has complex simulation model construction process and lack of information exchange with system entities [37]. Compared with it, CT ontology provides the formalization of HLA models to support the graphical construction of HLA federations, which greatly improves the efficiency of simulation development. This also provides a potential model-driven solution for SoS development and operation.

The CT is used to make decisions for SoS operations in the case study. First, the integrated simulations among the DTs are implemented to predict the performances of each UAV in different landing scenarios. Such simulation results are used as cues to develop ontology models in the CT. Through reasoning applied to the ontology models, one decision-making option representing a landing scenario with the least overall landing time is obtained. Based on this option, when the real SoS operates, the control center is able to manipulate each UAV for its landing target with the least landing time. Compared with traditional SoS development, the developed CT can use the simulation results to support decision-making during the real SoS operations, thereby connecting the virtual DTs to the real SoS operations.

6.3. *Summary*

The CT is an emerging concept that has been proposed by several studies [33,38]. However, there is a lack of basic frameworks to support CT development and applications, particularly for SoS development and operation [38,39]. In this study, we proposed a CT framework for decision-making for SoS operations based on DT integration and reasoning. The proposed framework uses the BFO and GOPPRRE ontologies to formalize real SoSs, their virtual models, and the topology between them. Moreover, through HLA specifications, heterogeneous DTs are integrated to implement simulations to predict the behaviors of each constituent system in the SoS. Finally, based on the simulation results, ontology models are used to provide reasoning results for decisions regarding the real SoS operations. From the case study, we found that the proposed CT approach enables decision-making for SoS operations with advanced features, which was confirmed by both quantitative and qualitative analyses.

7. Conclusion

This study proposes a CT to support DT integration and decision-making across systems when implementing SoS development and operation. For *challenge 1*, a semantic modeling approach is proposed to support a unified DT description in order to manage the complexity. The GOPPRRE approach supports the formalization of the systems, BFO and IOF specifications provide the integration of formalization of different domains to support complexity management. For *challenge 2*, semantic models are transformed into HLA execution models for integrated simulation among DTs for each constituent system in the real SoS through HLA-RTI. Through reasoning applied to the semantic models and simulation results, CT enables the support of SoS operations by providing decision-making options and solves *challenge 3*.

In conclusion, our research makes three main contributions:

- We proposed a new CT concept for supporting SoS development and operations. The CT employs semantic models to manage the complexity of DTs across systems by using unified descriptions.
- The CT provides an integrated simulation infrastructure for managing the interfaces between heterogeneous DTs and implementing data exchange among them.
- The CT offers cognitive capabilities based on semantic models and results from HLA simulations. Through reasoning applied to the semantic models, the CT enables decision-making support for real SoS operations.

From the case study, our approach indeed promotes complexity management capabilities for DTs across systems using a unified semantic model. Moreover, HLA is an infrastructure that supports integrated simulation across DTs, which promotes the interoperability of heterogeneous DTs via a unified interface specification. It allows SoS developers to construct DTs for their own systems without considering the limitations of heterogeneous data structures. Furthermore, reasoning based on semantic models and results from integrated simulations enables the provision of decision-making options for manipulating system behaviors of constituent systems during SoS operations. This provides a new model-based solution to construct SoSs for autonomous systems. In the future, a complete toolkit for ontology modeling, distributed simulation, and visualization for SoS development and operation will be developed to enable automatic distributed simulation execution, DT management, and decision-making support for real-time SoS operations.

Acknowledgments

This work was supported by The National Key Research and Development Program of China (Grant No. 2020YFB1708100). Thanks to the IoF Systems Engineering Working Group to support the ontology specification.

References

[1] Keating C, Rogers R, Unal R, Dryer D, Sousa-Poza A, Safford R, et al. System of systems engineering. EMJ – Engineering Management Journal. 2003; 15(3): 36-45.
[2] Zheng C, Eynard B, Qin XS, Li J, Bai J, Gomes S, et al. A requirement-driven architecture definition approach for conceptual design of mechatronic systems. Integrated Computer-Aided Engineering. 2019; 26(4): 361-82.
[3] Hause MC, editor. SOS for SoS: A new paradigm for system of systems modeling. 2014 IEEE Aerospace Conference; 1-8 March 2014.
[4] ISO/IEC/IEEE International Standard – Systems and software engineering – System of systems SoS) considerations in life cycle stages of a system. 2019: pp. 1-40.
[5] Piaszczyk C. Model based systems engineering with department of defense architectural framework. Systems Engineering. 2011; 14(3): 305-26.
[6] Force USA. United States Air Force Scientific Advisory Board Report on System of Systems Engineering for Air Force Capability Development, 2005.
[7] Törngren M, Grogan P. How to deal with the complexity of future cyber-physical systems? Designs. 2018; 2: 40.
[8] Wang XY, Zhang GX, Gou XT, Paul P, Neri F, Rong HN, et al. Multi-behaviors coordination controller design with enzymatic numerical P systems for robots. Integrated Computer-Aided Engineering. 2021; 28(2): 119-40.
[9] Zotov E, Tiwari A, Kadirkamanathan V. Conditional StyleGAN modelling and analysis for a machining digital twin. Integrated Computer-Aided Engineering. 2021; 28(4): 399-415.
[10] Zhang H, Wang J. An unsupervised semantic sentence ranking scheme for text documents. Integrated Computer-Aided Engineering. 2021; 28(1): 17-33.
[11] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, 2010: 1-38.
[12] Smith B, Grenon P. Basic Formal Ontology Available from: http://ontologybuffalo.edu/bfo.
[13] Kulvatunyou BS, Wallace E, Kiritsis D, Smith B, Will C, editors. The industrial ontologies foundry proof-of-concept project. IFIP WG 57 International Conference on Advances in Production Management Systems, APMS 2018, August 26, 2018 – August 30, 2018; Seoul, Korea, Republic of Springer Science and Business Media, LLC.
[14] Sheard SA, Mostashari A. Principles of complex systems for systems engineering. Systems Engineering. 2009; 12(4): 295-311.
[15] Unger A, Shean D, Silva F, Nguyen J, Beebe L, Dewire P, et al. Distributed architecture for monitoring urban air quality: A systems engineering approach. INCOSE International Symposium. 2020; 30: 620-36.
[16] Ring J, Madni AM, editors. Key challenges and opportunities in 'system of systems' engineering. 2005 IEEE International Conference on Systems, Man and Cybernetics; 12-12 Oct. 2005.
[17] Grieves M. Digital twin: Manufacturing excellence through virtual factory replication. White Paper. 2014; 1: 1-7.
[18] Tao F, Zhang M. Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing. IEEE Access. 2017; 5: 20418-27.
[19] Perez-Hurtado I, Martinez-del-Amor MA, Zhang GX, Neri F, Perez-Jimenez MJ. A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. Integrated Computer-Aided Engineering. 2020; 27(2): 121-38.
[20] Mirzaei G, Adeli H. Machine learning techniques for diagnosis of alzheimer disease, mild cognitive disorder, and other types of dementia. Biomedical Signal Processing and Control. 2022; 72: 103293.
[21] Song EY, Burns M, Pandey A, Roth T, editors. IEEE 1451 Smart Sensor Digital Twin Federation for IoT/CPS Research. 2019 IEEE Sensors Applications Symposium (SAS); 11-13 March 2019.
[22] Sun H, Fan W, Shen W, Xiao T, Chai Y. Ontology maintenance in high level architecture federation development and execution process. Integrated Computer-Aided Engineering. 2013; 20: 79-94.
[23] Hatledal LI, Skulstad R, Li G, Styve A, Zhang H. Co-simulation as a fundamental technology for twin ships. Modeling, Identification and Control. 2020; 41(4): 297-311.
[24] Martín F, Gomez-Donoso F, Escalona F, Rodríguez J, Cazorla M. Semantic visual recognition in a cognitive architecture for social robots. Integrated Computer-Aided Engineering. 2020; 27: 1-16.
[25] Zhang DJ, He FZ, Tu ZG, Zou L, Chen YL. Pointwise geometric and semantic learning network on 3D point clouds. Integrated Computer-Aided Engineering. 2020; 27(1): 57-75.
[26] Soylu A, Moedritscher F, De Causmaecker P. Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. Integrated Computer Aided Engineering. 2012; 19: 93-109.
[27] Kern H, Hummel A, Kühne S. Towards a comparative analysis of meta-metamodels. ACM International Conference on Sys-

tems, Programming, Languages, and Applications; Portland, Oregon, USA2011. pp. 7-12.

[28] Monticolo D, Hilaire V, Gomes S, Koukam A. A multi-agent system for building project memories to facilitate design process. Integrated Computer-Aided Engineering. 2008; 15.

[29] Rzevski G, Skobelev P, Zhilyaev A, Lakhin O, Mayorov I, Simonova E, editors. Ontology-Driven Multi-Agent Engine for Real Time Adaptive Scheduling. 2018 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO); 19-21 May 2018.

[30] Lu J, Wang G, Tao X, Wang J, Törngren M. A Domain-specific Modeling Approach Supporting Tool-chain Development with Bayesian Network Models. Integrated Computer Aided Engineering. 2019; 27.

[31] Kharlamov E, Martin-Recuerda F, Perry B, Cameron D, Fjellheim R, Waaler A, editors. Towards Semantically Enhanced Digital Twins. 2018 IEEE International Conference on Big Data (Big Data); 10-13 Dec. 2018.

[32] Rosen R, Boschert S, Sohr A. Next generation digital twin. Atp Magazin. 2018; 60: 86.

[33] Lu J, Zheng X, Gharaei A, Kalaboukas K, Kiritsis D. Cognitive Twins for Supporting Decision-Makings of Internet of Things Systems. 2020; pp. 105-15.

[34] Lu J, Ma J, Zheng X, Wang G, Li H, Kiritsis D. Design ontology supporting model-based systems engineering formalisms. IEEE Systems Journal. 2021; pp. 1-12.

[35] Wang H, Zhu S, Tang J, Lu J, Wu J, Kiritsis D, editors. Model-based Systems Engineering Supporting Cost Analysis of Aircraft Development Process. 2021 IEEE International Symposium on Systems Engineering (ISSE); 13 Sept.–13 Oct. 2021.

[36] Arp R, Smith B, Spear AD. Building ontologies with basic formal ontology. The MIT Press, 2015.

[37] Bocciarelli P, D'Ambrogio A, Giglio A, Paglia E, editors. Automated generation of fom modules for hla-based distributed simulations. 2019 Spring Simulation Conference (SpringSim); 29 April-2 May 2019.

[38] Abburu S, Berre AJ, Jacoby M, Roman D, Stojanovic L, Stojanovic N, editors. COGNITWIN – Hybrid and Cognitive Digital Twins for the Process Industry. 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC); 15-17 June 2020.

[39] Li X, He B, Wang Z, Zhou Y, Li G, Jiang R. Semantic-enhanced digital twin system for robot-environment interaction monitoring. IEEE Transactions on Instrumentation and Measurement. 2021; 70.