Journal of Industrial Information Integration 40 (2024) 100619

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.elsevier.com/locate/jii

Full length article

Design ontology for cognitive thread supporting traceability management in model-based systems engineering

Shouxuan Wu [a], Guoxin Wang [a], Jinzhi Lu [b,*], Zhenchao Hu [c], Yan Yan [a], Dimitris Kiritsis [d]

[a] Beijing Institute of Technology, Beijing 100081, China
[b] Beihang University, Beijing, 100191, China
[c] Shanghai Jiaotong University, Shanghai, 200240, China
[d] EPFL SCI-STI-DK, Station 9, CH-1015, Lausanne, Switzerland

ARTICLE INFO

Keywords:
Cognitive thread
Digital thread
Traceability management
Model-based systems engineering
Open service for lifecycle collaboration
KARMA language

ABSTRACT

Industrial information integration engineering (IIIE) is an interdisciplinary field to facilitate the industrial information integration process. In the age of complex and large-scale systems, model-based systems engineering (MBSE) is widely adopted in industry to support IIIE. Traceability management is considered the foundation of information management in MBSE. However, a lack of integration between stakeholders, development processes, and models can decrease the effectiveness and efficiency of the system development. A modified MBSE toolchain prototype has been developed to implement traceability management; however, a lack of formal and structured specifications makes it difficult to describe the complex topology in traceability management scenarios using this MBSE toolchain, such as creating traceability between heterogeneous models, which leads to poor reusability of this MBSE toolchain in other traceability management scenarios. To formalize traceability management scenarios using the MBSE toolchain, a cognitive thread (CT) ontology is developed in this study. The CT ontology is a specification expressing the information of stakeholders, models, and development processes for traceability management, providing the cognition capability to analyze the interrelationships between them. Based on the implementation of the modified MBSE toolchain, the concepts and interrelationships in the CT ontology are identified. The CT ontology is designed to develop the MBSE toolchain prototype for building, managing, and analyzing traceability in various traceability management scenarios. A case study of an adaptive cruise control system design is used to evaluate the completeness of the CT ontology through qualitative and quantitative analyses. The results demonstrate that the proposed CT ontology formalizes the information related to traceability management while using the proposed MBSE toolchain and can also be used in common traceability management scenarios to design other complex engineered systems.

1. Introduction

With the development of information and communication technology on the industry, the integration of multi-domain data sources wields more significant influences on complex product developments [1]. To facilitate the industrial information integration process, a newly emerging technical field industrial information integration engineering (IIIE) has been immense [2]. In literature, IIIE methodologies play a crucial role in solving all complex engineering problems of system analysis, design, optimization, validation and operation in different disciplines such as mathematics, computer science, aerospace engineering, etc [3–5]. Actually, IIIE may be perceived as a system engineering methodology to handle data source across myriad disciplines, which were separately handled with traditional information infrastructures [6]. Over the past two decades, the model-based systems engineering (MBSE) has been proposed and applied in economic systems, socio-technical systems (man-made systems), and engineering systems [7], which can advance and integrate the concepts, theory, and methods in each relevant discipline and establish a new discipline for industry information integration purposes as IIIE suggested [8].

The digital thread (DT) [9] is considered the next paradigm for information management and integration to realize MBSE, which can be seen as the framework linking information generated from all stages of the product lifecycle for decision-making [10]. In the implementation of MBSE, a variety of domain-specific modeling tools are used to develop models for formalizing system characteristics, such as system definition, analysis, and verification & validation (V&V) of complex

* Corresponding author.
E-mail address: jinzhl@buaa.edu.cn (J. Lu).

https://doi.org/10.1016/j.jii.2024.100619
Received 13 June 2022; Received in revised form 1 April 2024; Accepted 25 April 2024
Available online 30 April 2024

systems [11]. Due to increasing system scale, highly fragmented information and knowledge exchange exists across the lifecycle [12], causing unmanageable system complexity. Therefore, the traceability management between models throughout the lifecycle should first be considered to avoid losing valuable information and knowledge when establishing a DT. The traceability provides a unique authoritative data source, which serves as the basis for better decision making in the system engineering activities [13].

However, traceability management still faces challenges during DT implementation. The formalism of traceability management scenarios should first be considered as a prerequisite for traceability construction [14]. Traceability management depends on complex topological interrelationships in different system-development scenarios. Each scenario implements the design activities related to the model and data, development process, and stakeholders. For example, system engineers develop the physical architecture models, which are used by simulation engineers for the system-level verification of the design requirement. These models are completed in the corresponding design task, such as *system verification and validation*. Thus, the formalism of traceability management scenarios is required to provide unified representations that specify the semantic of concepts and the topological relationships between the related elements mentioned above.

In addition, during traceability management, the dependency between the model and data, development process, and stakeholders should be well managed and analyzed. Traceability management is the process of interacting, integrating, and accumulating existing knowledge for system development activities. The complicated development and configurations across various domains creates difficulty in handling the complex dependency of these elements. The model and data are created by stakeholders, who are typically from different domains and hierarchies. Well-managed dependency of the model and data across stakeholders promotes efficiency and effectiveness in the system development [15]. Furthermore, the sequence of the development process determines how the model and data in different sub-processes interact with each other [16]; thus, the dependency of the process, as well as the model and data, impacts the traceability management across the entire lifecycle. Therefore, increasing the ability to analyze traceability for the DT can improve the knowledge reusability, which promotes efficiency of the modeling or simulation. Furthermore, without sufficient data integration and tool interoperability, information exchange for traceability management between different domain tools is difficult [17,18]. The domain tools adopt proprietary data formats and data access protocols to develop tool application programming interfaces (APIs), models, and data. These heterogeneous APIs, models, and data result in barriers for communication and understanding of domain knowledge due to poor tool interoperability when implementing traceability management.

The main contribution of this study is the design of a cognitive thread (CT) ontology to support traceability management in an MBSE toolchain. Compared with other traceability management approaches, such as design structure matrix, the CT ontology provides a semantic description for the process of building and analyzing traceability, allowing computers to manage traceability automatically. In a previous study [19], we proposed the concept of CT, which serves as a DT with enhanced semantic ability, to formalize the interrelationship between the system artifacts, development process, and organization. The MBSE toolchain mentioned above utilizes a service-oriented approach to manage the topological interrelationships between stakeholders, development processes, and system artifacts, which is discussed in Section 3.2. The designed CT ontology aims to promote the efficiency of the MBSE toolchain development for different traceability management scenarios as follows: (1) formalize the concept and interrelationships between stakeholders, development processes, and system artifacts; (2) provide cognitive ability based on predefined reasoning rules. These rules are developed based on the interrelationships between stakeholders, development processes, and system artifacts. The cognitive ability helps stakeholders analyze and understand the complex dependency between the development process and the model and data in the co-design for selecting the optimal design.

In a previous study [20], an MBSE toolchain was developed to support co-design. To implement traceability management, we improve this toolchain as the basis of CT ontology formalism, which includes the following four aspects: (1) Open Services for Lifecycle Collaboration (OSLC) services of the system artifacts, e.g., domain-specific modeling (DSM), data, etc.; (2) web-based process management system (WPMS) generated from the DSM models; (3) unified authentication systems; and (4) OSLC service management system (OSMS). Based on the implementation of the MBSE toolchain, the concepts and interrelationships in the CT ontology are identified. Through a case study on the MBSE toolchain for adaptive cruise control (ACC) system design, the completeness of the CT ontology is evaluated through qualitative and quantitative measurements.

The remainder of this article is organized as follows. We discuss the related work in Section 2. In Section 3, we introduce our research design. In Section 4, the CT ontology is proposed in detail. A case study of an ACC system is adopted to validate the CT ontology in Section 5, and Section 6 evaluates the CT ontology through qualitative and quantitative analyses. Finally, the conclusions are offered in Section 7. The glossary for this article is shown in Table 1.

2. Related work

In this section, we first review the state of the art of traceability management in the MBSE domain, including concepts definition and ontology design. Then, the concepts of CT and current efforts to create a CT for traceability management are discussed. Finally, the challenges of traceability management in the MBSE domain are analyzed.

2.1. Concept and state of the art of traceability management

The IEEE standard glossary of software engineering terminology defines traceability as the "degree to which a relationship can be established between two or more products of the development process" [30]. Although there is no agreed-upon definition of traceability or traceability management in the MBSE field, the traceability concept for developing engineering systems is similar to that in software engineering [31]. The International Council on Systems Engineering (INCOSE) defines traceability in the MBSE field as "a discernible association between two or more logical entities such as requirements, system elements, verifications, or tasks" [32].

In the MBSE domain, engineers from different disciplines create various models to describe the system from different views and hierarchies. Explicitly defining and managing the interrelationships between models in the required granularity is a major endeavor for traceability management [33]. Traceability aims to trace and manage each design change during the product design phase [34], which facilitates the product and process data management, enabling the collection of useful information across all stages of the product lifecycle [35].

The design structure matrix is the most commonly used approach for traceability management [36]. Lee et al. [37] proposed a goal-driven-requirement traceability approach using a design structure matrix partitioned into blocks to analyze requirement change impacts. Sharon et al. [38] used the design structure matrix to manage the relationships between the project and product views in conceptual models based on object process methodology. R. Brahmi et al. [39] determined if all the parts of the assembly contribute to the satisfaction of the geometrical requirements by using SysML allocation matrix mechanical assembly design. D. Tang et al. [40] introduced design structure matrix (DSM) to realize the traceability between requirements and system architectures in system design. However, the software implementation of the design structure matrix relies on an additional tool set and is often limited by

Table 1
Glossary along with the definition.

Acronym	Description
DT [21]	Digital Thread, a technical concept frame based on authoritative source of truth, providing the ability of linking, accessing and integrating the data, the information and the knowledge.
CT [19]	Cognitive Thread, the concept of Digital Thread with enhanced semantic ability to formalize the interrelationship between model and data, development process and organization.
OSLC [22]	Open Service for Lifecycle Collaboration, a linked data technique for tool integration.
MBSE [23]	Model-Based Systems Engineering, a new technical tendency, which makes use of models to replace textual requirements to support system development.
FMI [24]	Functional Mock-up Interface, a specification of co-simulation interface.
GOPPRR-E [25]	Meta meta-models including Graph, Object, Point, Property, Relationship, Role and Extension, which includes the binding rules between these meta meta-models.
ACC [23]	Adaptive cruise control, an available cruise control advanced driver-assistance system for road vehicles that automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead.
OOSEM [26]	Object-Oriented Systems Engineering Method, an MBSE modeling methodology.
WPMS [20]	a Web-based Process Management Platform that generated from BPM Camunda.
RBAC [27]	Role-Based policies Access Control, an approach to restricting system access to authorized users.
DSM [28]	Domain-Specific Modeling, an engineering approach to describe facets of system abstractions through models built based on domain-specified modeling language (DSML) to describe information for specific domains and systems through a set of formal meta-models.
BPMN [29]	Business Process Modeling Notation, a specification for process modeling.

the tools; thus, an additional cost is incurred when using the design structure matrix to create traceability links between different tools.

Model transformation is another basic traceability management approach between specific tools. Guo et al. [41] designed code-generation language syntax to realize the traceability between MBSE and Simulink models. Kharrat et al. [42] extended a systems modeling language (SysML) profile and designed a model transformation approach to realize traceability between SysML and computer-aided design (CAD) models. However, most of the model transformation approaches only support unidirectional information transformations and are insufficient for the bidirectional information transformations required in some traceability management scenarios.

Additionally, a toolchain [43] has been defined as a set of DSM tools to perform complex system development tasks, which is considered a potential solution for traceability management in an MBSE domain. Lu et al. [44] proposed an MBSE toolchain to support automated parameter value selection for co-simulation, which facilitates the traceability between the simulation configuration and development process. The definition of an MBSE toolchain is one toolchain consisting of two or more modeling, simulation, and design tools that, when combined, can support and construct a system engineering workflow [45]. Didem et al. [46] discussed a node-link diagram (NLD) visualization technique for visualizing interoperability in cyber–physical system (CPS) development toolchains. Jad et al. [47] proposed an approach for integrating structured product data residing in conventional databases of a software toolchain based on linked data technology. The INTO-CPS project provided a toolchain approach for integrated verification based on MBSE and co-simulation approaches [48]. OpenMETA is another toolchain that transforms DSM models into simulation models for design works, including architecture design and verification [49]. DSM is an engineering approach to describe information for systems in specific domains through models and a set of formal meta-models. These toolchains promote the traceability between models in different domains. In this study, we focus on the MBSE toolchain development for supporting traceability management. To facilitate data exchange and promote semantic interoperability between different tools in the MBSE toolchain, the ontology design for the MBSE toolchain is also considered, including the composition of MBSE toolchains, etc.

2.2. Ontology design for traceability management in MBSE

In the MBSE domain, ontology refers to the types of objects and their properties and relationships, which represent domain-specific knowledge [50]. Ontology is used to formalize system information of system development and artifacts in MBSE practice. For example, Lu et al. [51] developed the ontology to formalize an MBSE toolchain for co-simulation. Hennig et al. [52] used ontology to develop a space system. Chen et al. [53] developed the ontology for behavior verification in the system design stage. To accurately describe traceability, ontology is also widely used to construct elements and their topological relationships in traceability management. Nitesh et al. [54] used ontology to extract the textual and structural information in the meta-model of software artifacts using MBSE, e.g., bugs and use cases. Antony et al. [55] used ontology to establish traceability between the abstract requirements and architectural design specifications of software. However, most of the formalism only focuses on representing the entities and relationships of the traceability management scenarios, rather than the process of implementing traceability management. Therefore, such formalism lacks the capability to support automated operations traceability management scenarios, such as building traceability.

Due to the inherent features of engineered systems, the iterations and interactions between various design activities and stakeholders also add to the complexity of the product realization processes [56]. Therefore, the development process information must be included in the designed ontology. Practical efforts have been made in some previous studies using ontology to formalize the development process. Wang et al. [57] proposed a phase-event-information-X (PEI-X) ontology for meta-design process hierarchies for a heat exchanger. However, most of these formalisms only focus on the descriptions of the development processes themselves; the relationships between process information and traceability information are ignored.

2.3. Cognitive thread for traceability management

The DT was first proposed for F-35 fighter jet development [58]. As digital engineering strategies [59] evolve, the concept of DT is constantly updated and expanded: T. D. West et al. [60] considered it

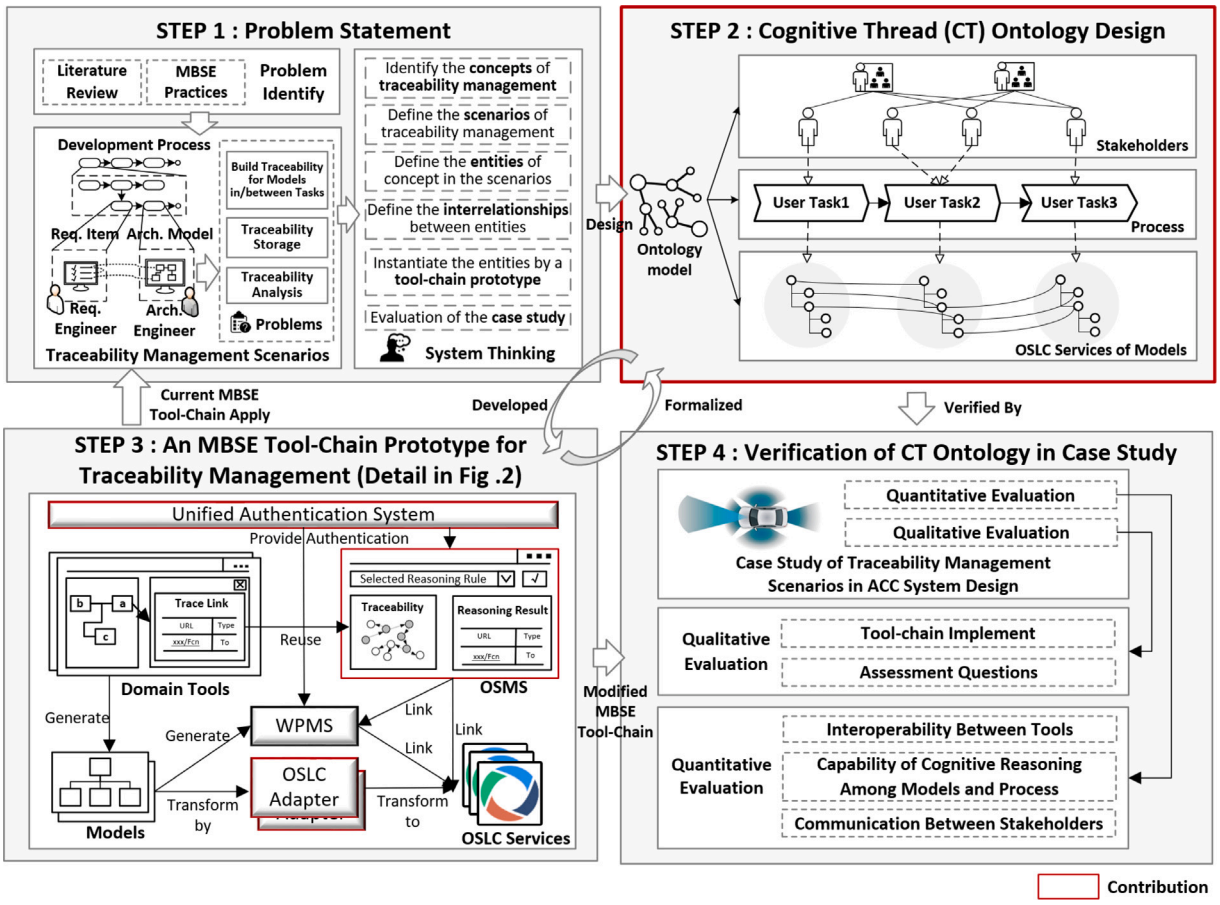

Fig. 1. Overview of research design.

as "a framework for merging the conceptual models of the system with the discipline-specific engineering models of various system elements". Kraft et al. [21] considered it as "an extensible, configurable, and enterprise level framework that seamlessly expedites the controlled interplay of authoritative data, information, and knowledge to inform decisions during a system lifecycle by providing the capability to access, integrate, and transform disparate data into actionable information". Recently, researchers have attempted to enhance the DT with cognitive capabilities using semantic technologies. The paradigm of the next-generation DT was proposed as CT [19], which serves as a DT with enhanced semantic ability to formalize the interrelationship between the model and data, development process, and organization.

To realize the analysis and intelligent reasoning for traceability, semantic technologies have been applied to a CT. Ontology and knowledge graphs have been created by researchers to describe the topological interrelationship between digital models. Bone et al. [61] proposed an interoperability and integration framework based on ontology and provided the ability of design information retrieval based on a semantic model and ontology modeling language. Hedberg et al. [62] extended the method of linking data in a distributed manufacturing system and built a DT based on the distributed object identifier and knowledge graphs. Kwon et al. [63] and Helu et al. [64] used knowledge graphs and mechanical standards such as STEP, EXPRESS, and QIF to build the DT, realizing semantic mapping between format data and ontology, which integrates the model and data in the manufacturing and inspection stages; this ontology is used to share the semantics of terms or models in different domains to improve data interoperability and traceability in the CT. However, the lack of formalizing the CT framework leads to difficulties for developing an MBSE toolchain based on the CT concept to implement traceability management.

2.4. Summary

Based on literature reviews, we can summarize several key motivations related to our work. (1) From Section 2.1, to formalize the MBSE toolchain for supporting traceability management, the ontology to describe the MBSE toolchain itself should be developed, such as the compositions and their interrelationships in the MBSE toolchain. (2) From Section 2.2, unlike current studies from academia and industry, the ontology needs to not only formalize the development process and system artifacts but also support process management and tool operations, particularly traceability management. (3) From Section 2.3, various semantic technologies have been developed to enhance the knowledge expression ability in traceability management using the CT. However, the existing methods do not include the analysis of interrelationships between the models, development process, and stakeholders. Thus, semantic reasoning ability is required to support traceability analysis based on the ontology.

3. Research design

The research design of this study is depicted in Fig. 1. In STEP 1, the traceability management scenarios are identified based on the challenges of traceability management from the literature review and MBSE practices. System thinking is adopted to support the design of the CT ontology in STEP 2, which solves the problems in the application of the current MBSE toolchain. Based on the CT ontology, the current MBSE toolchain is modified to support traceability management scenarios in STEP 3. Finally, the CT ontology is evaluated quantitatively and qualitatively by the toolchain prototype used in the case study of the ACC system design, which is shown in STEP 4.

3.1. Problem statement

In [20,51], we formalized and developed the MBSE toolchain to automatically implement aero-engine simulation. Stakeholders use the DSM models to formalize the development process and system artifacts. Then, the DSM models are transformed into one WPMS to deploy technical resources and implement automated co-simulations in Simulink for different stakeholders.

However, some difficulties exist using the current MBSE toolchain to solve traceability management problems, as shown in STEP 1 of Fig. 1. First, although this toolchain supports building traceability between a work task and related technical resources, it cannot support building traceability between these technical resources. Second, for different work tasks in the development process, this toolchain cannot support building traceability between the related technical resources of these tasks. Finally, traceability storage and analysis mechanisms for stakeholders are not provided in the current toolchain. For example, the mechanism for deciding whether stakeholders can access traceability information is not provided.

In this study, systems thinking is utilized to analyze the design of the CT ontology to improve the current MBSE toolchain to support traceability management. Systems thinking is an approach for understanding systems by examining the interactions between the components within the system boundary [65]. Thus, the problem statement for this study consists of the following steps:

(1) *Define the concept of traceability and traceability management in the MBSE toolchain*: Based on the literature and MBSE practices, the concepts of traceability and traceability management are defined to specify the purposes for improving the MBSE toolchain. In this study, we define the concept of traceability and traceability management based on the definitions in [19], which are as follows:

- *In the MBSE domain,* ***traceability*** *is the ability to define a bidirectional relationship between system artifacts, e.g., models, documents, codes, etc., based on the inherent relationships between the development processes.*
- ***Traceability management*** *is the ability to build, maintain, acquire, and analyze the traceability to achieve the understanding of system artifacts among stakeholders within development processes through specific methods, such as reasoning and visualization.*

(2) *Define the scenarios of traceability management in the MBSE toolchain*: Several scenarios of traceability management are defined to specify the scope of the contents (systems boundary) of the improved MBSE toolchain. Each scenario is used to define a function of the improved MBSE toolchain, which satisfies the operational process related to traceability management. For example, the traceability between technical resources (two relevant requirement items) related to a specific work task can be built (detailed in Section 4.4).

(3) *Define the entities related to the scenarios*: In each scenario, the related entities are captured to describe the composition and function of the scenario. For example, the model (requirement item or block of the simulation model) and work tasks are included in the scenario mentioned above.

(4) *Define the interrelationships between entities*: Interrelationships between entities refer to the connections between entities, such as the trace relationship between a requirement item and a block of the simulation model included in the scenario mentioned above.

(5) *Instantiate the entities using a toolchain prototype*: Based on the entities and interrelationships mentioned above, a toolchain prototype is developed to support traceability management scenarios (detailed in Section 3.2).

(6) *Evaluation of the case study*: Through the case study of the ACC system design, qualitative and quantitative methods are proposed to evaluate the developed toolchain prototype (detailed in Section 3.3).

3.2. Prototype strategy for cognitive thread ontology design

Based on the CT ontology, we develop an MBSE toolchain prototype[1] to support traceability management scenarios, thereby improving the toolchain proposed in [19,20]. Fig. 2 shows an overview workflow of the service-oriented toolchain prototype. Typically, stakeholders use domain tools to generate models and data, formalizing the development process and system artifacts, as shown in Fig. 2(a). These models and data are transformed into one WPMS to implement the co-design and several OSLC services for data inter-operation, as shown in Figs. 2(b) and (c). Then, an OSMS is developed to capture the OSLC services of models, as shown in Fig. 2(e). In the OSMS, the traceability between the OSLC services of the models and data, as well as work tasks in the WPMS, are built. Several reasoning rules are developed based on the traceability management scenarios, which help to analyze the traceability and provide an understanding of the complex topological relationships among the domain knowledge. Additionally, through the access control of the unified authentication system shown in Fig. 2(d), the traceability information is provided to the appropriate stakeholders, which enhances consistent understanding of the design information. The compositions of the developed MBSE toolchain prototype are used to formalize the concepts and relationships in the CT ontology, as shown in Fig. 1(a).

As shown in Fig. 2(a), we adopt three domain tools to develop the models and data to implement the system design. Excel[2] is used to develop the requirement table for extracting the structured stakeholder needs. Based on these stakeholder needs, MetaGraph[3] is used to build DSM models to describe the system architectures. These DSM models are built based on the GOPPRR-E modeling approach and KARMA modeling language[4] (introduced in Section 4.1). The DSM models include process, system architecture, and simulation models. The process models are used to formalize the development process. The system architecture models are used to represent the related information in each work task, including composition, interface, behavior, and parameters of the system. The simulation models are used to represent the topology for system validation and verification. Using the code generators in MetaGraph, the DSM models are transformed into the ontology to describe the development processes and co-simulation model structures. Based on this ontology, Simulink[5] is used to developed a simulation model and execute co-simulation.

Subsequently, OSLC adapters are developed to transform these heterogeneous models and data into OSLC services through specific mapping rules, as shown in Fig. 2(b). The detailed mapping rules are illustrated in Section 4.1. In the OSLC adapters, the structures of the models and data are analyzed, and key information is extracted using the data parser. This information is used to generate uniform resource identifiers (URIs) of the OSLC services, which are addresses for finding details of the OSLC services. Additionally, resource description framework (RDF) information of the OSLC services is generated, which provides a semantic description for the OSLC services. Based on the RDF information and linked data, the interrelationships between related OSLC services are discovered and generated automatically through a representational state transfer (REST) API.

In addition, the process models in the DSM models are loaded by the compiler to generate a WPMS where each work task is linked to the corresponding OSLC services, as shown in Fig. 2(c). The developers log into their own accounts in the WPMS and access the required information and technical resources for each work task through the

[1] Video available at: https://www.youtube.com/watch?v=Okg3zBDJMIQ&t=1324s.

[2] https://www.microsoft.com/zh-cn/microsoft-365/microsoft-office?rtc=1

[3] http://www.zkhoneycomb.com/

[4] A modeling language based on the GOPPRR-E modeling approach.

[5] https://ww2.mathworks.cn/products/simulink.html

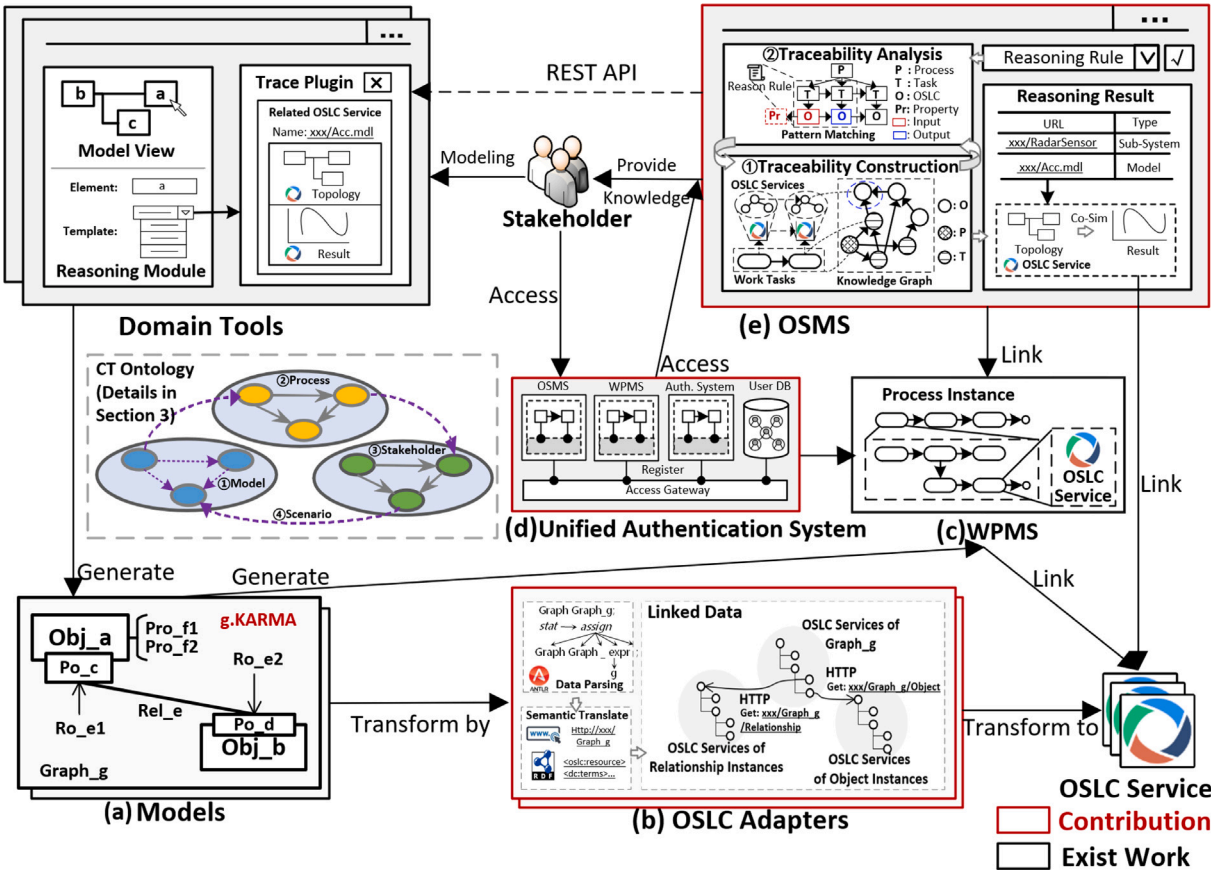

Fig. 2. Overview of the MBSE toolchain prototype.

URI of related OSLC services. Using the WPMS, the system co-design is automatically implemented.

The unified authentication system is also used to provide data access control, as shown in Fig. 2(d), and it includes an access gateway, authority system, and user database. The user database consists of several permission tables that store user accounts and the API permissions. Once the authentication starts, the authority system validates the stakeholder's credentials with the user database and determines whether to allow access to the API. All applications in this MBSE toolchain are registered to the access gateway to orchestrate the API execution sequence.

Finally, the OSMS is developed to store and analyze the traceability between the OSLC services and development process in the WPMS, as shown in Fig. 2(e). The OSMS uses a Neo4j[6] database to store the traceability relationships in several knowledge graphs. All the work tasks, sub-processes, and processes in the WPMS are mapped to the *Task* and *Process* nodes, while all the OSLC services are mapped to the *OSLC* nodes in the knowledge graphs, as shown in Fig. 2(e)①. The detailed properties of the nodes and edges in the Neo4j database are illustrated in Table 2 and are used to design the topological relationships in the CT ontology. Base on the traceability management scenarios, several scenario-based reasoning rules based on the Cypher language [66] are developed, as shown in Fig. 2(e)②. Traceability management includes the following four scenarios: (1) the scenario of managing the traceability between OSLC services, (2) the scenario of managing the affiliations between OSLC services and corresponding work tasks, (3) the scenario of managing the stakeholder operations to manipulate the OSLC services, and (4) the scenario of managing the allocation relationships between work tasks and stakeholders. Based on these scenarios, the reasoning engine in the OSMS specifies the pattern of the reasoning rules. By matching the specific pattern in the Cypher language, the reasoning results are provided as a basis for decision making in the development process. To improve the design efficiency, several tool plugins are developed to reuse the reasoning rules and provide a graphical interface to visualize the reasoning results.

The implementation of the MBSE toolchain described above satisfies the requirements stated in Section 3.1. However, to migrate this toolchain for more complex traceability management scenarios, the information and inter-relationships between the system artifacts, development process, and stakeholders should be formalized. Therefore, a CT ontology is suggested to formalize the MBSE toolchain to support traceability management, which is detailed in Section 4.

3.3. Evaluation based on case study

A case study of an ACC system design is proposed to validate the feasibility and effectiveness of the MBSE toolchain prototype, which is developed based on the proposed CT ontology. In the case study, qualitative and quantitative measurements are considered to evaluate the efficiency of the MBSE toolchain prototype, as shown in Table 3.

To obtain the qualitative measurements, the following two approaches are adopted:

(1) *Toolchain Prototype Development*: To validate whether the CT ontology supports describing the logic in traceability management scenarios between models, development processes, and stakeholders, four traceability management scenarios are defined as metrics to validate the MBSE toolchain as follows:

(a) *Traceability management scenario between OSLC services of models*: refers to a situation where different OSLC services

[6] https://neo4j.com/

Table 2
The properties of nodes and edges in traceability knowledge-graph.

Node			Edge		
Type	Property	Description	Type	Property	Description
OSLC	Name	Name for nodes of OSLC services.	OSLCRelationships <OSLC,OSLC>	Type	Describing the relationship between different OSLC services, such as *trace*.
	URI	Corresponding URI of OSLC services.	OSLCInTask <OSLC,Task>	Type	Describing the affiliated relationship between OSLC services and development process, such as *belongTo*.
	Tool	Tools that generating models.	HaveProcess <Process,Process>	Type	Describing the affiliated relationship between different development process, such as process has a sub-process.
Process	Name	Name for nodes of development processes.	HaveTask <Process,Task>	Type	Describing the affiliated relationship between development process and work tasks, such as *sub-process has a userTask*.
	Type	Identify the process is process or sub-process	TaskLinkToProcess <Task,Process>	Type	Describing the connection relationship between development process and work tasks, such as *a task links to a subprocess with sequence flow*.
Task	Name	Name for nodes of work tasks in development processes.	TaskLinkToTask <Task,Task>	Type	Describing the connection relationship between different work tasks, such as *a task links to a subprocess with sequence flow*.

Table 3
Measurement and metrics for evaluation.

Measurement	Views	Metrics	Evaluation approach
Qualitative	Logic in traceability management scenarios	Managing traceability between OSLC services of models Managing traceability between OSLC services of models & Development Processes Managing traceability between OSLC services of models & Stakeholders Managing traceability between Development Processes & Stakeholders	Toolchain prototype
	Efficiency of trace-ability management	Level of automation for the generation of trace links Storage of trace links Mechanism for Visualizing (representing) trace links Creation, deletion, modification and/or querying of trace links Interchange of traceability information Analysis with the traceability information obtained	Assessment questions
Quantitative	Interoperability of models	OSLC services generated OSLC services type	Statistic
	Levels of communication between stakeholders in development processes	Stakeholders participating in development processes Role of stakeholders Permission of stakeholders Project that stakeholders belong to Tasks in each development process	
	Capability for cognitive reasoning to analyze traceability	Traceability links established in knowledge graphs Reasoning templates Query result type	

of models are connected in specific relationships, which are illustrated in Section 4.4. For example, the OSLC service of the requirement block in the DSM model traces to the OSLC service of the row in the requirement table.

(b) *Traceability management scenario between OSLC services of models and development processes*: refers to a situation where OSLC services of models and data are linked to corresponding work tasks. For example, the text of the requirement block can be modified by the OSLC service in a work task.

(c) *Traceability management scenario between OSLC services of models and stakeholders*: refers to a situation where the OSLC services of model elements cannot be accessed if the permission information of the stakeholder is invalid. For example, the project manager cannot access the OSLC services of the DSM models.

(d) *Traceability management scenario between development processes and stakeholders*: refers to a situation where the work tasks are allocated to a specific stakeholder to finish. For example, the work task of specifying stakeholder requirements is allocated to the project manager.

(2) *Assessment Questions Evaluation*: Seven assessment questions are proposed in [67] for scoring the traceability management approaches in the model-driven engineering field. In this study, we adopt these assessment questions to evaluate our toolchain:

(a) *Does it suggest or consider any techniques to improve the level of automation for the generation of trace links?* Scores: Y

(Yes), the methodological proposal does provide ideas or techniques on how to automate the generation of traces; P (Partly), it suggests techniques to generate traces semi-automatically; N (No), the proposal does not consider automating the generation of traces.

(b) *Does it suggest a technique for the storage of trace links?* Scores: Y, it suggests mechanisms to materialize trace links in some form of data structure. Two options are mainly considered: external traceability, required from database schemes or meta-models, and internal traceability, where trace links are collected in the models themselves through textual references or hyperlinks; P, it considers the storage of trace links or acknowledges that it has to be supported, but does not provide any mechanism to do so; N, the proposal does not consider the storage of trace links.

(c) *Does it suggest a mechanism for visualizing (representing) trace links?* Scores: Y, it suggests the development of an ad hoc tool to support the visualization of trace links or some other graphical representation, such as a traceability matrix; P, it leans on generic tools to display trace links, instead of using tools devised to represent this type of information, and the resulting representation cannot be optimal or intuitive; N, the proposal does not suggest any form of representing trace links.

(d) *Does it consider the creation, deletion, modification, and/or querying of trace links? If so, does it state the way in which such tasks should be supported?* Scores: Y, the proposal considers the four tasks; P, the proposal does not consider all of these tasks, only some of them; N, the proposal does not consider any of the tasks.

(e) *Does it provide ideas or techniques for the interchange of traceability information between different proposals?* Scores: Y, the proposal addresses the problem of traceability information interchange and offers solutions; P, the proposal only identifies the interchange problem, but it does not offer any solution; N, the proposal does not consider the interchange of traceability information.

(f) *Does it perform some type of analysis with the traceability information obtained?* Scores: Y, the proposal aims to analyze traceability information with the goal of providing refined information to the different stakeholders; P, the proposal identifies information obtained from the traces that needs to be classified or analyzed, but it does not provide a method of doing so; N, the proposal does not consider the analysis of traceability information.

When using a quantitative approach, we are inspired by the SPIRIT framework [68] and, therefore, measure three views of the efficiency of our toolchain by defining the following metrics:

(a) *Interoperability of Models*: refers to the capability of multi-domain models to exchange information with each other. In the case study, this is measured by the metrics of the number of OSLC service entities that are generated in total.

(b) *Levels of Communication between Stakeholders in Development Processes*: refers to the capability of stakeholders to access the models and communicate with each other. In the case study, this is measured from process and stakeholder perspectives. From the process perspective, it is measured by the number of process instances and work tasks in each process. From the stakeholder perspective, it is measured by the number of projects, stakeholders in each project, roles, and permissions of each stakeholder.

(c) *Capability of Cognitive Reasoning to Analyze Traceability*: refers to the capability of analyzing the traceability relationships between nodes of the OSLC service and work tasks by reasoning in knowledge graphs. In the case study, this is measured by the number of traceability links that are established in the knowledge graphs and the total numbers of reasoning templates and reasoning result types.

4. Ontology of cognitive thread for traceability management in MBSE

In this section, we introduce the CT ontology for designing the MBSE toolchain in Fig. 3, which corresponds to STEP 2 in Section 2. The CT ontology includes the following four parts.

(a) The ontology formalizing models is developed based on the OSLC services [22] of the models, which are detailed in Section 4.1. These models represent information and characteristics of the target systems, such as requirements and system architecture.

(b) The ontology formalizing the development process is developed based on process models in the Business Process Model and Notation (BPMN) [29] specification, which is detailed in Section 4.2. The workflow in the process models represents the relationships of work tasks in the real-world development process. The development process refers to the task sequence of designing a target system.

(c) The ontology formalizing stakeholders is developed based on the development of a unified authentication system, which is detailed in Section 4.3. The unified authentication system is developed based on the role-based access control (RBAC) model [27], which is a permission control model for allocating specific permissions to stakeholders. Stakeholder refers to an individual, team, or organization with interests in, or concerns related to, the target systems.

(d) The ontology formalizing the scenarios of traceability management is developed based on topological relationships between development processes, models, and stakeholders, which is detailed in Section 4.4. In this study, we focus on four traceability management scenarios: (1) Traceability management scenarios between OSLC services of models; (2) Traceability management scenarios between OSLC services of models and development processes; (3) Traceability management scenarios between OSLC services of models and stakeholders; (4) Traceability management scenarios between development processes and stakeholders.

The cognition capability of the CT is developed as the reasoning process to analyze these relationships in the traceability management scenarios. In the reasoning process, the domain questions from stakeholders are analyzed, classified, and formalized as rules to provide reasoning results. These results are used to provide domain knowledge to stakeholders as the decision-making foundation in the system development process.

4.1. Ontology formalizing models

In the MBSE domain, models are the most common system artifacts for formalizing different systems. Based on the various degrees of formalism, models can be divided into formal and informal models [69]. Based on the relationship types of the elements in the model, the formal models can be further divided into descriptive and analytical models [69,70]. In the proposed MBSE toolchain, the structured requirement documents are considered informal models, the DSM models are descriptive models, and the Simulink models are analytical models.

For example, as shown on the left side of Fig. 4, the DSM models are generated based on the GOPPRR-E modeling approach [25] because it is a powerful approach to describe domain-specific characteristics with better descriptive capabilities than other meta-modeling approaches. The GOPPRR-E meta-meta-models include the *Graph*, *Object*, *Point*, *Property*, *Role*, *Relationship*, and *Extension*, as follows.

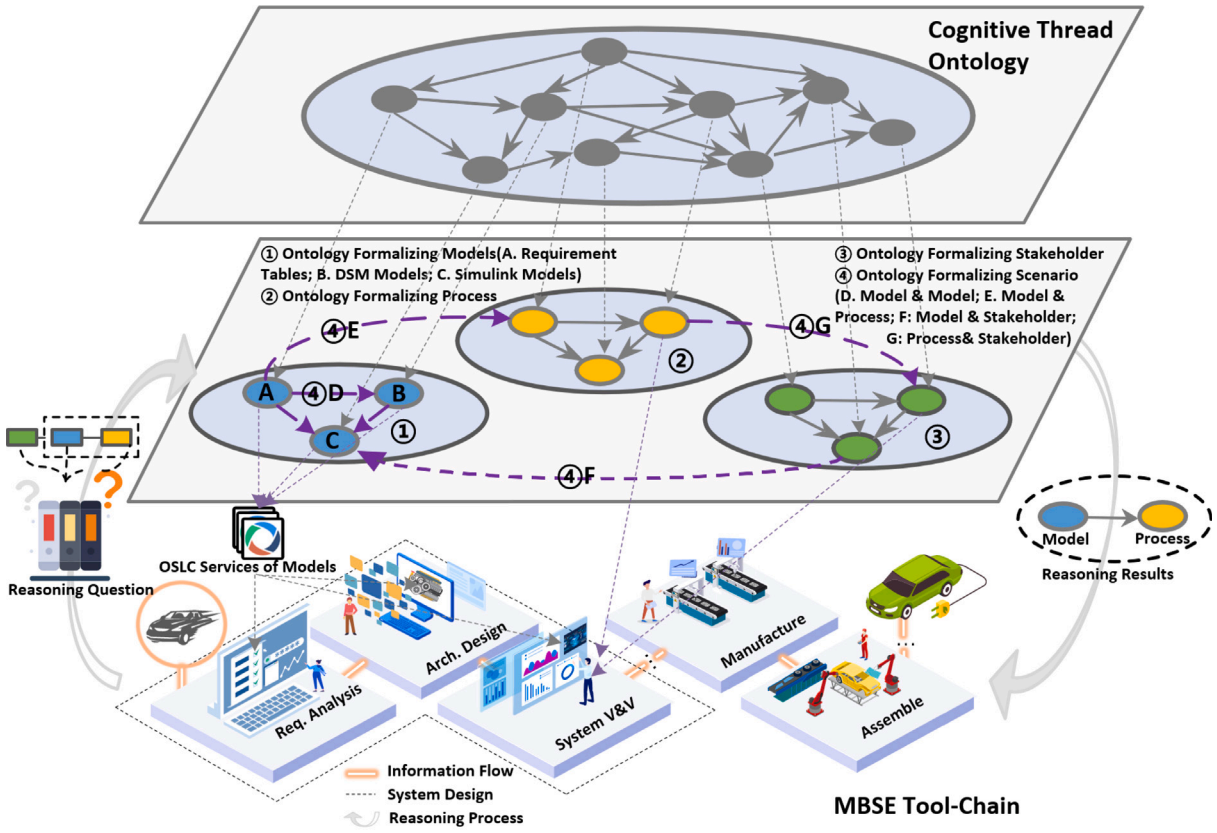

Fig. 3. CT ontology for traceability management.

(a) *Graph*: a collection of the *Object, Relationship*, and *Role* represented as one window (one integrated concept of a class diagram and package in the unified modeling language (UML)). The *Graph* can be a visual diagram on the top level or lower level decomposed by one *Object*.
(b) *Object*: one entity in the *Graph* (class concepts in UML).
(c) *Point*: a port in the *Object*.
(d) *Relationship*: one connection between the different *Points* of the *Object*.
(e) *Role*: used to define the connection rules mirrored to the relevant *Relationship*. For example, one *Relationship* has two *Roles*, each of which is defined to connect with one *Point* in the *Object*. Then, the *Relationship* is connected with these *Points* in the *Object*.
(f) *Property*: one attribute of the other five meta-meta-models.
(g) *Extension*: the additional constraints used to construct meta-models, such as a connector for binding one *Point* on the *Object* and one *Role* on one side of the *Relationship*.

In the proposed MBSE toolchain, the OSLC services of the requirement tables, DSM models, and Simulink models are developed to support data integration and inter-operation based on OSLC core specification [22]. These OSLC services describe the compositions and properties of the models, which are then used to design the ontology formalizing models. The mapping process between the OSLC services and models is shown in Fig. 4, and the model compositions of the DSM models are mapped to concepts in the OSLC core model, such as *ServiceProviderCatalog*, *ServiceProvider*, *Service*, and *Resource*, as follows.

(a) *ServiceProviderCatalog*: a list used in the discovery of OSLC *ServiceProviders*. In DSM models, the model library of model instances is considered a *ServiceProviderCatalog*.
(b) *ServiceProvider*: a container that provides an implementation of *Services*. In DSM models, the model instances are considered *ServiceProviders*. A *ServiceProvider* offers information for viewing the link to a *Resource* and rich previews of the *Resource*.
(c) *Service*: a set of capabilities that enable a web client to manage *Resources*. In DSM models, the APIs for operating model instances are considered *Services*, such as the APIs for creating, retrieving, updating, and deleting the instances of GOPPRR-E meta-models.
(d) *Resource*: an atomic object structure that is configured to be consumed by the OSLC, such as the instances of GOPPRR-E meta-models. Attributes of the objects in the object structure are mapped to the RDF as predicates for the *Resource* and specify the namespaces for the types of *Resources*.

For example, as shown on the right side of Fig. 4, the model instance *G.KARMA* is mapped into the *ServiceProvider* concept and transformed into the RDF to describe the model instance information. Based on the linked data principle, the *ServiceProvider* of *G.KARMA* links to the *Resource* of the graph instance *Graph_g*. Furthermore, the *Resource* of *Graph_g* links to the related *Resources* of other instances of the GOPPRR-E element, such as instances of object *Obj_a* and relationship *Rel_e*.

In this study, ontology is used to formalize the OSLC services of models and the interrelationships between the OSLC services. To define the ontology, we use the partial ontology representation in the Web Ontology Language (referred to as OWL) [71] for reference to construct our ontology. Several transformation rules from OSLC services of models to OWL concepts are developed, based on OSLC core specifications: (1) *ServiceProviderCatalog*, *ServiceProvider*, *Service*, and *Resource* are transformed into hierarchical *Class* concepts in OWL. (2) The interrelationships of concepts are defined as *object properties* in OWL. Details of *object properties* are listed in the following tables, which can be read as <row item, action, column item>. For example, the $<Graph_a^{(id,t)}$, *include*, $Object_b^{(id,t)}>$ refers to the OSLC service representing $Graph_a^{(id,t)}$ that includes the OSLC service representing $Object_b^{(id,t)}$.

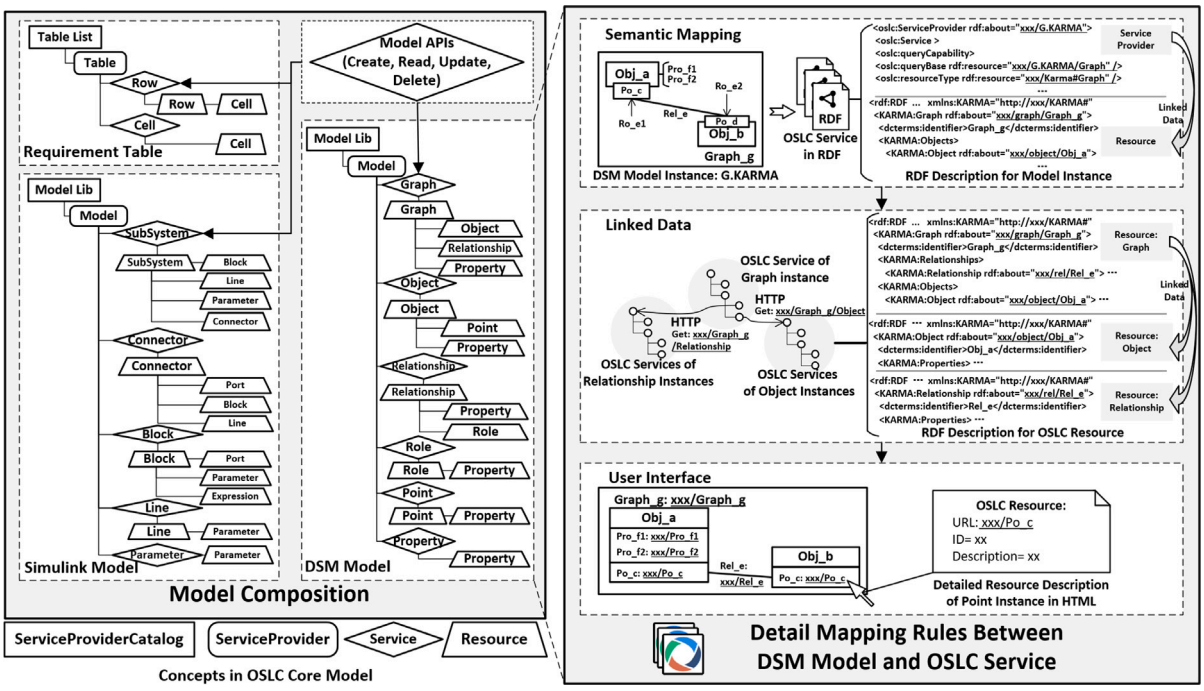

Fig. 4. Mapping process between OSLC services and models.

(3) The property values of *ServiceProviderCatalog*, *ServiceProvider*, *Service*, and *Resource* are defined as data property assertions in OWL. For example, the property *URL* of the OSLC *Resource Object* links to the data property of the OSLC *Resource Object1* with assertions "URL".

Definition 1. The symbol ::= refers to a collection of ontology concepts. The symbol => refers to the object property between two given ontology concepts.

Thus, the ontology of the models is formalized using the OSLC services for requirement tables, DSM models, and Simulink models in the MBSE toolchain as follows:

$$O\_Model ::= \{O\_ReqTable_a, O\_DsmModel_b, O\_SimuModel_c\} \quad (1)$$

O_Model refers to the ontology of the OSLC services for the collection of requirement tables, DSM models, and Simulink models, as shown in Fig. 3①. $O\_ReqTable_a$ refers to ath ontology of the OSLC services for structured requirement tables, as shown in Fig. 3(a)①. These requirement tables allow engineers to communicate using spreadsheets or written specifications in a table, document, or database. $O\_DsmModel_b$ refers to bth ontology of the OSLC services for DSM models, as shown in Fig. 3(b)①. These DSM models represent stakeholder knowledge or understanding about logical relationships within target systems. $O\_SimuModel_c$ refers to cth ontology of the OSLC services for Simulink models, as shown in Fig. 3(c)①. These Simulink models represent relationships through mathematical equations and are used to execute co-simulation.

In Fig. 3(a)①, the structured requirement tables are used to formalize stakeholder needs in the proposed MBSE toolchain. Therefore, the ontology of the OSLC services for requirement tables is given based on the composition and contents of the elements in the requirement tables as follows:

$$O\_ReqTable_i ::= \left\{TableList_a^{type}, Table_b^{type}, Row_c^{type}, Cell_d^{type}\right\} \quad (2)$$

$O\_ReqTable_i$ refers to the ith ontology of the OSLC service for the requirement table, $TableList_a^{type}$ refers to the ath OSLC service for the set of requirement tables, $Table_b^{type}$ refers to the bth OSLC service of the requirement table, Row_c^{type} refers to the cth OSLC service of the row in requirement tables, $Cell_d^{type}$ refers to the dth OSLC service of the cell in requirement tables, and the superscript $type$ indicates the type of OSLC service of the related ontology element, such as *ServiceProviderCatalog*, *ServiceProvider*, *Service*, and *Resource*. Details of the object properties between individuals of $O\_ReqTable_i$ are illustrated in Table 4.

Table 4
Object properties between individuals of $O\_ReqTable_i$.

=>	$TableList_a^{type}$	$Table_b^{type}$	Row_c^{type}
$Table_b^{type}$	include		
Row_c^{type}		include	
$Cell_d^{type}$		include	include

In Fig. 3(b)①, the DSM models are developed to formalize the development processes and system characteristics, e.g., requirement, in the proposed MBSE toolchain. Therefore, based on the composition and relationships in the DSM models, the ontology of the OSLC services for DSM models is given as follows:

$$O\_DsmModel_i ::= \left\{ DsmLib_g^{(id,t)}, DsmModel_h^{(id,t)}, Graph_a^{(id,t)}, Object_b^{(id,t)}, Property_c^{(id,t)}, Point_d^{(id,t)}, Relationship_e^{(id,t)}, Role_f^{(id,t)} \right\} \quad (3)$$

$O\_DsmModel_i$ refers to the ith ontology of the OSLC service for the DSM model. $DsmLib_g^{(id,t)}$ refers to the gth OSLC service for sets of the DSM models. $DsmModel_h^{(id,t)}$ refers to the hth OSLC service of the DSM models. $Graph_a^{(id,t)}$ refers to the ath OSLC service of the *Graph* concept in GOPPRR-E, which is a collection of *Object*, *Point*, *Property*, *Relationship*, and *Role* represented together as a visual diagram and can be decomposed by an *Object*. $Object_b^{(id,t)}$ refers to the bth OSLC service of the *Object* concept in GOPPRR-E, which is one entity in a *Graph*. $Property_c^{(id,t)}$ refers to the cth OSLC service of the *Property* concept in GOPPRR-E, which is an attribute of the other five meta-meta-models. $Point_d^{(id,t)}$ refers to the dth OSLC service of the *Point* concept in GOPPRR-E, which is a port in the *Object*. $Relationship_e^{(id,t)}$ refers to the eth OSLC service of the *Relationship* concept in GOPPRR-E, which is one connection between the different *Points* or *Objects*.

Table 5
Object properties between individuals of $O\_DsmModel_i$.

=>	$Graph_a^{(id,t)}$	$Object_b^{(id,t)}$	$Point_d^{(id,t)}$	$Relationship_e^{(id,t)}$	$Role_f^{(id,t)}$	$DsmLib_g^{(id,t)}$	$DsmModel_h^{(id,t)}$
$DsmModel_g^{(id,t)}$						include	
$Graph_a^{(id,t)}$		decomposite explosion		explosion	explosion		include
$Object_b^{(id,t)}$	include						include
$Point_d^{(id,t)}$		include			connect		include
$Relationship_e^{(id,t)}$	include						include
$Role_f^{(id,t)}$				startFrom endTo			include
$Property_c^{(id,t)}$	include	include	include	include	include		include

Table 6
Object properties between individuals of $O\_SimuModel_i$.

=>	$SimuModel_a^{(id,t)}$	$Block_b^{(id,t)}$	$Port_f^{(id,t)}$	$Line_c^{(id,t)}$	$Connector_g^{(id,t)}$	$SimuLib_h^{(id,t)}$	$Subsystem_d^{(id,t)}$
$SimuModel_a^{(id,t)}$						include	decomposite
$Block_b^{(id,t)}$	include				linkTo linkFrom		include
$Line_c^{(id,t)}$	include				linkTo linkFrom		include
$Subsystem_d^{(id,t)}$	include	include			linkTo linkFrom		include
$Parameter_e^{(id,t)}$	include	include	include	include			include
$Port_f^{(id,t)}$		include			linkTo linkFrom		
$Connector_g^{(id,t)}$	include						include

$Role_f^{(id,t)}$ refers to the fth OSLC service of the *Role* concept in GOPPRR-E, which is used to define the connection rules between the *Object* and the relevant *Relationship*. The superscript *id* refers to the instance types of the ontology concepts (meta-model), such as the SysML requirement diagram (one meta-model based on the *Graph*), which is an instance type of the *Graph*. The superscript t indicates the type of OSLC service of the related ontology element, such as *ServiceProviderCatalog* (spc), *ServiceProvider* (sp), *Service* (ser), and *Resource* (re). For example, $Graph_{RequirementDiagram1}^{(RequirementDiagram,sp)}$ refers to one OSLC *ServiceProvider* generated from the *RequirementDiagram1 Graph* in a descriptive model. Details of the object properties between individuals of $O\_DsmModel_i$ are illustrated in Table 5.

In Fig. 3(c)①, the Simulink models are developed to support system verification and validation in the proposed MBSE toolchain. Therefore, based on the composition and relationships in the Simulink models, the ontology of the OSLC services for the analytical model is given as follows:

$$O\_SimuModel_i ::= \left\{ SimuLib_h^{(id,t)}, SimuModel_a^{(id,t)}, Block_b^{(id,t)}, Line_c^{(id,t)}, Subsystem_d^{(id,t)}, Parameter_e^{(id,t)}, Port_f^{(id,t)}, Connector_g^{(id,t)} \right\} \quad (4)$$

$O\_SimuModel_i$ refers to the ith ontology of the OSLC service for the Simulink models. $SimuLib_h^{(id,t)}$ refers to the hth OSLC service for the sets of Simulink models. $SimuModel_a^{(id,t)}$ refers to the ath OSLC service of the Simulink model, which is the collection of *Block*, *Line*, *Subsystem*, *Parameter*, *Port*, and *Connector*. $Block_b^{(id,t)}$ refers to the bth OSLC service of the block in the Simulink models. $Line_c^{(id,t)}$ refers to the cth OSLC service of the line in the Simulink models, which is used to describe the connection between the different ports or blocks. $Subsystem_d^{(id,t)}$ refers to the dth OSLC service of the subsystem in the Simulink models, which is the set of blocks, lines, parameters, and ports for realizing the system sub-functions. $Parameter_e^{(id,t)}$ refers to the eth OSLC service of the parameters in the Simulink models, which are a set of values that determines the characteristics of the systems. $Port_f^{(id,t)}$ refers to the fth OSLC service of the port in the Simulink models, which is used as the interface on a block or line for external connections. $Connector_g^{(id,t)}$ refers to the gth OSLC service of the connector in the Simulink models, which is used as an interface for internal connections of the models. For example, $Block_{Out1}^{(Out,sp)}$ refers to one OSLC *ServiceProvider* generated from the *Out*1 block in the Simulink model. Details of the object properties between individuals of $O\_SimuModel_i$ are illustrated in Table 6.

4.2. Ontology formalizing development process

In the proposed MBSE toolchain, a WPMS is generated from the process models, which are part of the DSM models, and developed based on BPMN specifications to implement the co-design process for stakeholders. The BPMN-based process models can describe the actual development process. Thus, as Fig. 3② shows, we adopt the ontology definition in [68], which defines the ontology of the development process based on the composition and relationships in the process models as follows:

$$O\_Process ::= \left\{ BPMN\_Process_j \right\} \quad (5)$$

$$BPMN\_Process_j ::= \left\{ Process_i, Start_a, End_b, UserTask_c, Gateway_d, Sequence_e, AutoTask_f \right\} \quad (6)$$

$O\_Process$ refers to the ontology of the development process. $BPMN\_Process_j$ refers to the jth process model developed based on BPMN specifications. $Process_i$ refers to the ith process that includes several development activities. $Start_a$ refers to the ath atomic start event in a process. End_b refers to the bth atomic end event in a process. $UserTask_c$ refers to the cth atomic development task that is implemented by the user in a process. $Gateway_d$ refers to the dth atomic gateway used for decision making in a process. $Sequence_e$ refers to the eth sequence relationship connecting with the previous atomic elements in the process. $AutoTask_f$ refers to the fth work task that is automatically executed by the WPMS. Details of the object properties between individuals of $O\_Process$ are illustrated in Table 7.

4.3. Ontology formalizing stakeholders

In the proposed MBSE toolchain, a unified authentication system is developed based on RBAC-based permissions tables in a database to provide permissions management for stakeholders. The users and their

Table 7
Object properties between individuals of $O\_Process$.

=>	$Process_i$	$Start_a$	End_b	$UserTask_c$	$Gateway_d$	$AutoTask_f$
$Start_a$	include					
End_b	include					
$UserTask_c$	include					
$Gateway_d$	include					
$Sequence_e$	include	startFrom	toEnd	startFrom/ toEnd	startFrom/ toEnd	startFrom/ toEnd
$AutoTask_f$	include					

Table 8
Object properties between individuals of $O\_Stakeholders$.

=>	$Project_a$	$Stakeholder_b$	$Role_c$	$Permission_d$
$Stakeholder_b$	include			
$Role_c$	include	assignRole		
$Permission_d$	include	ownPermission	assignPermission	
API_e^t	include			operate

operation related to the unified authentication system are considered in the scope of stakeholders related to the MBSE toolchain. Thus, as Fig. 3③ shows, based on the composition and relationships in the RBAC-based permissions tables, the ontology of the stakeholders related to the development processes is defined as follows:

$$O\_Stakeholders ::= \left\{Project_a, Stakeholder_b, Role_c, Permission_d, API_e^t\right\} \tag{7}$$

$O\_Stakeholders$ refers to the ontology of the stakeholders involved in the related development process. $Project_a$ refers to the ath project in the organization for collaborative development. $Stakeholder_b$ refers to the bth stakeholder who directly participants in the development process and is interested or concerned about the system. $Role_c$ refers to the cth type of role for the stakeholders in the project. $Permission_d$ refers to the dth type of permission, which is provided by the roles in the project, for operating the API to access the development information. API_e^t refers to the eth type of API to access development information in the project, where the superscript t refers to the types, including: ① create the model or model composition from OSLC services; ② update the model or model composition from OSLC services; ③ retrieve the model or model composition from OSLC services; ④ delete the model or model composition from OSLC services; ⑤ execute the simulation; and ⑥ access the results of simulation. Details of the object properties between individuals of $O\_Stakeholders$ are illustrated in Table 8.

4.4. Ontology formalizing scenarios of traceability management

In the proposed MBSE toolchain, four scenarios of traceability management are identified: (a) between OSLC services of models; (b) between OSLC services of models and development processes; (c) between OSLC services of models and stakeholders; and (d) between development processes and stakeholders. As shown in Fig. 3④, to formalize the ontology of the traceability management scenarios, the topological links between development processes, models, and stakeholders are captured as follows:

$$O\_Scenario ::= \left\{O\_Scenario_{OO}, O\_Scenario_{OP}, O\_Scenario_{OS}, O\_Scenario_{PS}\right\} \tag{8}$$

$O\_Scenario$ refers to the ontology of traceability management scenarios between development processes, models, and stakeholders. $O\_Scenario_{OO}$ refers to the ontology of traceability management scenarios between the OSLC services of models. $O\_Scenario_{OP}$ refers to the ontology of traceability management scenarios between the OSLC services of models and development processes. $O\_Scenario_{OS}$ refers to the ontology of traceability management scenarios between the OSLC services of models and stakeholders. $O\_Scenario_{PS}$ refers to the ontology of traceability management scenarios between development processes and stakeholders.

Definition 2. The symbol $\in$ refers to a *belong to* relationship between ontology concepts. For example, $A \in \{A_1, A_2\}$ means that A can be either A_1 or A_2.

(1) *Traceability management scenario between OSLC services of models*: In this scenario, several types of traceability links between different OSLC services of models are identified. In the proposed MBSE toolchain, the relevant OSLC services of models are connected by traceability links with specific semantics that correspond to the relationship *OSLCRelationships<OSLC,OSLC>* in Table 2. As shown in Fig. 3(d)④, the ontology of this scenario consists of the following traceability links:

$$O\_Scenario_{OO} ::= \left\{TraceLink_i^{(t,linkS,linkT)}\right\} \tag{9}$$

$$linkSource = linkTarget \in \left\{O\_ReqTable_i, O\_DsmModel_j, O\_SimuModel_k\right\} \tag{10}$$

$TraceLink_i^{(t,linkS,linkT)}$ refers to ith traceability link between related OSLC services, where the superscript t refers to the type of link between OSLC services in accordance with MBSE practice, including: ① *trace to*: such as a requirement block traced to a corresponding requirement item; ② *link to*: such as a requirement block linked to a corresponding test case; ③ *verified by*: such as a requirement item verified by a corresponding simulation model block; ④ *transform to*: such as an internal block transformed into a corresponding simulation model block by code generation. The superscripts $linkS$ and $linkT$ refer to the start and end of $TraceLink_i^{(t,linkS,linkT)}$, respectively. $O\_ReqTable_i$, $O\_DsmModel_j$, and $O\_SimuModel_k$ refer to the corresponding ith/jth/kth OSLC service of the requirement table, DSM model, and Simulink model, respectively, which are considered the source and target of $Link_a^t$.

(2) *Traceability management scenario between OSLC services of models and development processes*: In this scenario, the links between the development processes and related OSLC services of models are captured. In the proposed MBSE toolchain, the work tasks in the WPMS link to the corresponding OSLC services to operate models for co-design, which corresponds to the relationship *OSLCInTask<OSLC,Task>* in Table 2. These work tasks are generated from the object instance of *UserTask* in the process model. Thus, as shown in Fig. 3(e)④, the ontology of this scenario is formalized based on the relationship between OSLC services and work tasks in the WPMS, which are defined as follows:

$$O\_Scenario_{OP} ::= \left\{ProcessLink_i^{(processLinkS,processLinkT)}\right\} \tag{11}$$

$$processLinkS \in \left\{O\_ReqTable_i, O\_DsmModel_j, O\_SimuModel_k\right\} \tag{12}$$

$$processLinkT \in \left\{UserTask_c\right\} \tag{13}$$

$ProcessLink_i^{(processLinkS,processLinkT)}$ refers to ith process link between the *UserTask* and related OSLC services. $processLinkS$ and $processLinkT$ refer to the start and end of $ProcessLink_i^{(processLinkS,processLinkT)}$, respectively. $O\_ReqTable_i$, $O\_DsmModel_j$, and $O\_SimuModel_k$ refer to the corresponding ith/jth/kth OSLC service of the requirement table, DSM model, and Simulink model, respectively. $UserTask_c$ refers to the cth work task in the WPMS.

(3) *Traceability management scenario between OSLC services of models and stakeholders*: In this scenario, two types of links between stakeholders and OSLC services of models in the MBSE toolchain are identified, as seen in Fig. 2. ① the links representing the stakeholders operate the traceability links between OSLC services, such as the engineer creating a traceability link; ② the links representing operation of OSLC services, such as the engineer querying an OSLC service. Therefore, as shown in Fig. 3(f)④, the ontology of this scenario is defined as follows:

$$O\_Scenario_{OS} ::= \left\{ Operation_i^{(ty,operationS,operationT)} \right\} \tag{14}$$

$$operationS \in \left\{ API_d^t \right\} \tag{15}$$

$$operationT \in \left\{ TraceLink_c^{(t,linkS,linkT)}, O\_ReqTable_i, O\_DsmModel_j, O\_SimuModel_k \right\} \tag{16}$$

$Operation_i^{(ty,operationS,operationT)}$ refers to ith operation of the stakeholder on OSLC services or the traceability link between them, where the superscript ty refers to the type of operation, including: ① create; ② update; ③ retrieve; and ④ delete. $operationS$ and $operationT$ refer to the start and end of $Operation_i(ty, operationS, operationT)$, respectively. API_d^t refers to the dth API for operating the OSLC services by stakeholders. $TraceLink_c^{(t,linkS,linkT)}$ refers to the cth traceability link of OSLC services that is manipulated. $O\_ReqTable_i$, $O\_DsmModel_j$, and $O\_SimuModel_k$ refer to the corresponding ith/jth/kth OSLC service of the requirement table, DSM model, and Simulink model that are manipulated, respectively.

(4) *Traceability management scenario between development processes and stakeholders*: In this scenario, the links between the development processes and related stakeholders are captured. In Section 4.2, the work tasks in the WPMS are allocated to different stakeholders in the unified authentication system of the proposed MBSE toolchain. Thus, as shown in Fig. 3(h)④, the ontology of this scenario is formalized based on the associations between work tasks in the WPMS and stakeholders in the unified authentication system, defined as follows:

$$O\_Scenario_{PS} ::= \left\{ Allocation_i^{(allocationS,allocationT)} \right\} \tag{17}$$

$$allocationS \in \left\{ UserTask_c \right\} \tag{18}$$

$$allocationT \in \left\{ Stakeholders_b \right\} \tag{19}$$

$Allocation_i^{(allocationS,allocationT)}$ refers to ith allocation relationship between the stakeholder in the MBSE toolchain and the work task in the WPMS. $allocationS$ and $allocationT$ refer to the start and end of the allocation relationship, respectively. $UserTask_c$ refers to the cth work task in the WPMS. $Stakeholders_b$ refers to the bth stakeholder in the MBSE toolchain.

Details of the object properties between the individuals of the scenarios for traceability management are listed in Table 9.

5. Case study

5.1. Scenario definition

ACC is a collision avoidance system that adapts the conventional control based on the vehicle's external driving environment, such as the speed and distance of the vehicle ahead. To verify the completeness of the proposed CT ontology, a case study of ACC system design [23] is used to evaluate the MBSE toolchain prototype.

As shown in Fig. 5(a), the process of ACC system design starts with the Excel tool. Project managers capture top-level stakeholder requirements to develop the ACC system, as shown in Fig. 5(b). Then, based on these stakeholder requirements, the DSM models of the ACC system are developed by system engineers in MetaGraph 2.0, as shown in Fig. 5(c)①②③. These DSM models formalize the system characteristics, such as system requirements. Based on the DSM models, the Simulink models are developed by simulation engineers to execute co-simulation based on functional mock-up interface (FMI) specification for system-level verification, as shown in Fig. 5(d). The CarSim software is utilized to develop the ACC scenario and generate functional mock-up units (FMUs) for co-simulation. Simulink is used to develop the ACC system controller model, which is considered a master model of FMI-based co-simulation. The work flow of ACC system design is passed from Excel to MetaGraph 2.0 to Simulink, as shown in Fig. 5(b)(c)(d).

To confirm the completeness of the CT ontology and realize the traceability management between requirement tables, DSM models, and Simulink models, several case study scenarios are proposed based on the metrics in Table 3 as follows:

(1) *Scenario of managing traceability between OSLC services*: Supporting operations, such as building, deleting, changing, and querying the traceability between compositions of requirement tables, DSM models, and Simulink models (based on the metric *Managing Traceability Between OSLC Services of Models* in Table 3).
(2) *Scenario of managing traceability between OSLC services and work tasks*: Supporting operations, such as automatically building and querying the traceability between tasks in the process and models, including requirement tables, DSM models, and Simulink models (based on the metric *Managing Traceability Between OSLC Services of Models and Development Processes* in Table 3).
(3) *Scenario of analyzing traceability*: Providing reasoning capability for the following given domain problems to analyze traceability: ① querying a single OSLC service based on its name; ② querying a single OSLC service based on the tool name for creating it; ③ querying any directly related OSLC services based on their names; ④ querying any directly related OSLC services based on the sequence of corresponding work tasks; ⑤ querying any relevant OSLC services for the specific model element or data in the case study (based on the metrics *Managing Traceability Between OSLC Services of Models and Development Processes* and *Managing Traceability Between OSLC Services of Models* in Table 3).
(4) *Scenario of managing traceability between stakeholders and work tasks*: Supporting operations to models, such as querying the model topology, in specific work tasks in the development processes for stakeholders (based on the metric *Managing Traceability Between Development Processes and Stakeholders* in Table 3).
(5) *Scenario of managing traceability between OSLC services and stakeholders*: Supporting access control of models according to the permissions of stakeholders (based on the metric *Managing Traceability Between OSLC Services of Models and Stakeholders* in Table 3).

5.2. Models and data supporting ACC system design

The top-level stakeholder requirements are recorded in Excel, as seen in Fig. 5(b), and the Excel table corresponds to $Table_b^{type}$ in Eq. (2). The stakeholder requirements are captured in the table rows (Row_c^{type} in Eq. (2)), which include qualitative requirements, such as the abilities of driving assistance, speed controlling, and distance controlling. The stakeholder requirements also include quantitative requirements, such as the value of safe driving distances. The text of the stakeholder requirement, e.g., *the safe driving distance between two vehicles is not less than 45 m*, corresponds to $Cell_d^{type}$ in Eq. (2). As discussed in Section 3.2, the DSM models are developed based on the stakeholder requirements to support the formalism of the ACC system. The DSM models consist of three types: process models to describe the development processes (Fig. 5(c)①), system architecture models to describe

Table 9
Object properties between individuals of scenarios.

=>	$O\_ReqTable_i$	$O\_DsmModel_j$	$O\_SimuModel_k$	$UserTask_c$	API_j	$TraceLink_c$[a]	$Stakeholder_b$
$TraceLink_i$[b]	*linkTo/ linkFrom*	*linkTo/ linkFrom*	*linkTo/ linkFrom*				
$ProcessLink_i$[c]	*linkTo*	*linkTo*	*linkTo*	*linkFrom*			
$Operation_i$[d]	*linkFrom*	*linkFrom*	*linkFrom*		*linkTo*	*linkFrom*	
$Allocation_i$[e]				*linkTo*			*linkFrom*

[a] Here $TraceLink_c$ refers to the $TraceLink_c^{(t,linkS,linkT)}$.
[b] Here $TraceLink_i$ refers to the $TraceLink_i^{(t,linkS,linkT)}$.
[c] Here $ProcessLink_i$ refers to the $ProcessLink_i^{(processLinkS,processLinkT)}$.
[d] Here $Operation_i$ refers to the $Operation_i^{(ty,operationS,operationT)}$.
[e] Here $Allocation_i$ refers to the $Allocation_i^{(allocationS,allocationT)}$.

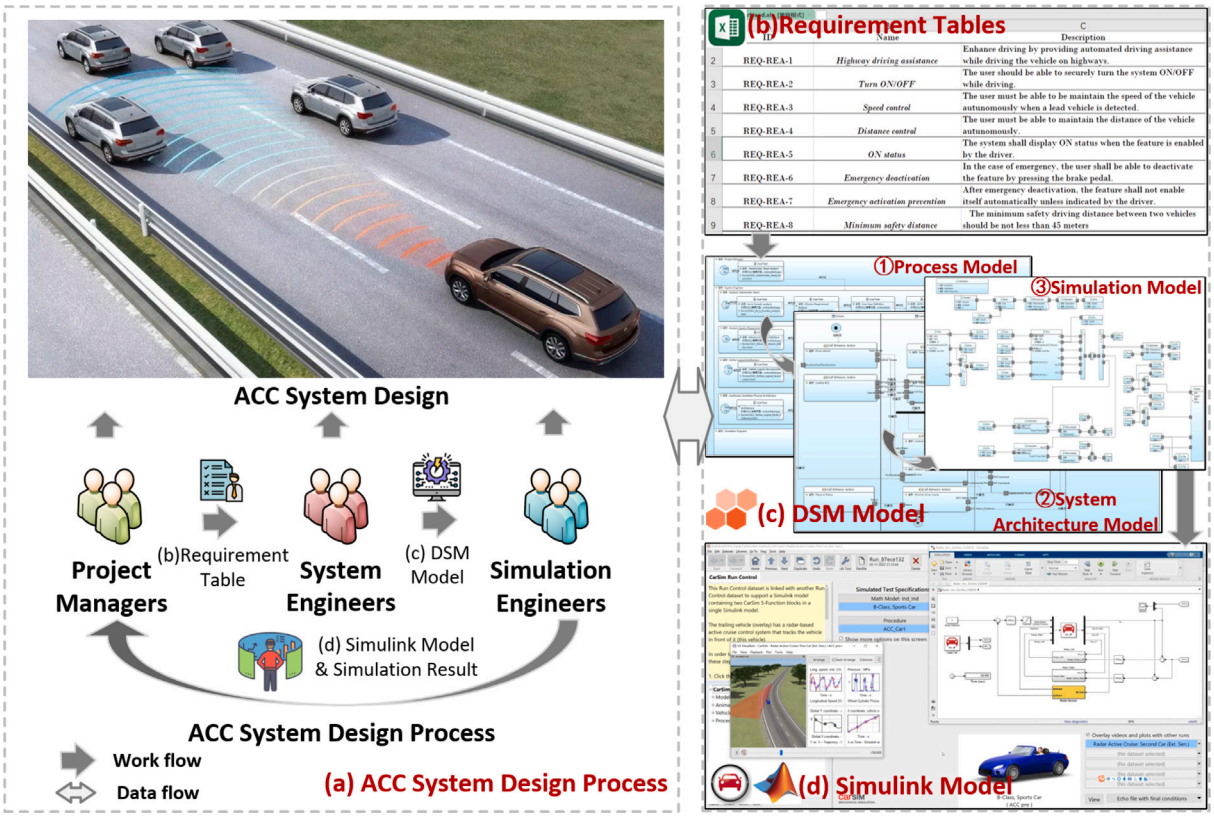

Fig. 5. ACC system design process.

the system architecture, such as composition and interfaces of the ACC system (Fig. 5(c)②), and simulation models to describe the Simulink model structure (Fig. 5(c)③). These DSM models are developed based on the KARMA language and GOPPRR-E modeling approach, as shown in the meta-model in Fig. 6.

(1) Fig. 6(a) shows the model dependency between process models, system architecture models, and simulation models. Each object instance *Work Task* includes graph instances *Simulink Model Structure* and *System Architecture Model*.
(2) Fig. 6(b) shows the meta-models of the process model. *Process* refers to a meta-model graph representing a process and corresponds to $Process_i$ in Eq. (6). In each process, there are five types of objects and one relationship: ① *Start* refers to the meta-object representing the start node of the process and corresponds to $Start_a$ in Eq. (6). ② *End* refers to the meta-object representing the end node of the process and corresponds to End_b in Eq. (6). ③ *UserTask* refers to the meta-object representing a stakeholder task and corresponds to $UserTask_c$ in Eq. (6). The *formKey* is a meta-property that links to a related OSLC service in *UserTask* and corresponds to $ProcessLink_i^{(processLinkS,processLinkT)}$ in Eq. (11). *Stakeholder* is a meta-property that allocates a *UserTask* to a specific stakeholder and corresponds to $Allocation_i^{(allocationS,allocationT)}$ in Eq. (17). ④ *ServiceTask* refers to the meta-object representing an automatic task executed by the WPMS and corresponds to $AutoTask_f$ in Eq. (6). *ClassName* is a meta-property that links to a java class to execute a *ServiceTask*. ⑤ *Gateway* refers to the meta-object representing a decision-making point during the development process and corresponds to $GateWay_e$ in Eq. (6). ⑥ *Sequence* refers to the meta-relationship that connects work tasks in the process and corresponds to $Sequence_e$ in Eq. (6).
(3) Fig. 6(c) shows sections of the meta-models of the system architecture models based on the semantic and syntax of SysML [72], including nine types of meta-graphs that correspond to $Graph_a^{(id,t)}$ in Eq. (3). Fig. 6(c)① shows part of the meta-graph of the requirement diagram (REQ) and the meta-objects and meta-relationships in REQ, which correspond to $Object_b^{(id,t)}$ and $Relationship_e^{(id,t)}$, respectively, in Eq. (3). Figs. 6(c)②–⑦ show the meta-objects and meta-relationships in the meta-graphs of the

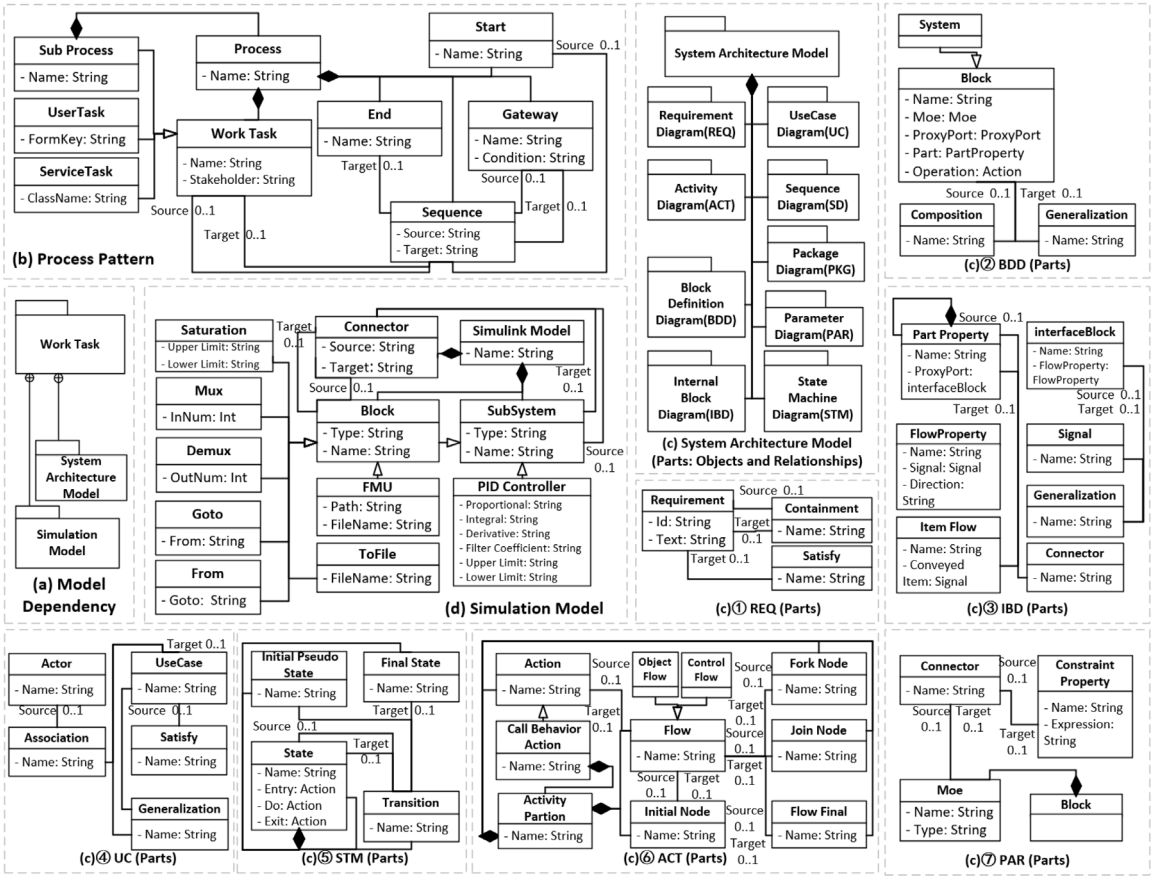

Fig. 6. The Meta-Model of DSM Models.

block definition diagram (BDD), internal block diagram (IBD), use case diagram (UC), state machine diagram (STM), activity diagram (ACT), and parameter diagram (PAR), respectively. These meta-objects and meta-relationships correspond to $Object_b^{(id,t)}$ and $Relationship_e^{(id,t)}$ in Eq. (3), respectively. Due to space constraints, we will explain the implications of all these meta-models in subsequent studies.

(4) Fig. 6(d) shows the meta-models of the simulation models, where the meta-graph, meta-object, meta-property, meta-point, meta-relationship, and meta-role correspond to $Graph_a^{(id,t)}$, $Object_b^{(id,t)}$, $Property_c^{(id,t)}$, $Point_d^{(id,t)}$, $Relationship_e^{(id,t)}$, and $Role_f^{(id,t)}$ in Eq. (3), respectively. As shown in Fig. 6(d), *Simulink Model* refers to a meta-graph representing a model for simulation. In each *Simulink Model*, *Block* refers to the meta-object representing a Simulink block. Specifically, *FMU* refers to the meta-object representing the FMU block in the Simulink model. *ToFile* refers to the meta-object representing a block for sending simulation data to a specific file. *Mux* and *Demux* refer to the meta-objects representing blocks for combining or decomposing signals, respectively. *From* and *Goto* refer to the meta-objects representing blocks for passing and receiving signals, respectively. *Saturation* refers to the meta-object representing a block for imposing upper and lower bounds on a signal. *SubSystem* refers the meta-object representing a set of blocks. *PID Controller* refers to the meta-object representing a proportional–integral–derivative (PID) controller subsystem. *Connector* refers to the meta-relationship that connects blocks or subsystems in the Simulink model.

Based on these meta-models, DSM models are developed, as shown in Fig. 7. All the meta-model instances of graph, object, property, point, relationship, and role correspond to $Graph_a^{(id,t)}$, $Object_b^{(id,t)}$, $Property_c^{(id,t)}$, $Point_d^{(id,t)}$, $Relationship_e^{(id,t)}$, and $Role_f^{(id,t)}$ in Eq. (3), respectively.

(1) Fig. 7(a) shows the instances of the process models. Three types of stakeholders are included in the process models: project managers, system engineers, and simulation engineers. Every *UserTask* in the process model can be linked and visualized by a detailed graph instance. For example, the *UserTask* mission requirement is linked to a requirement diagram, as shown in Fig. 7(b)①, to describe the mission requirements of the ACC system.

(2) Figs. 7(b)①–⑦ show part of the instances for the system architecture models. These model instances are developed based on the Object-Oriented Systems Engineering Method (OOSEM) modeling methodology [26]. Fig. 7(b)① shows the mission requirements specified for developing the ACC system. For example, the mission requirement *Driving assistance* is developed as the instance of the meta-object *Requirement* with the instance of the meta-properties *name* and *text*. Based on these requirements, the operation domain of the ACC system is developed, which is used to identify the scope of the ACC system in Fig. 7(b)②. For example, the *ACC System* is developed as the instance of the meta-object *System*. The block *ACC system* is used as a boundary of use cases in Fig. 7(b)③. For example, *Provide ACC* is developed as the instance of the meta-object *UseCase*. In Fig. 7(b)③, the activity diagram is developed in the use case *Provide ACC* in Fig. 7(b)⑤ to analyze the operation scenario of the ACC system. For example, *Calculate relative parameters* is developed as the instance of the meta-object *Call Behavior Action* with the instance of the meta-point *Host Vehicle Speed*. Based on the operation

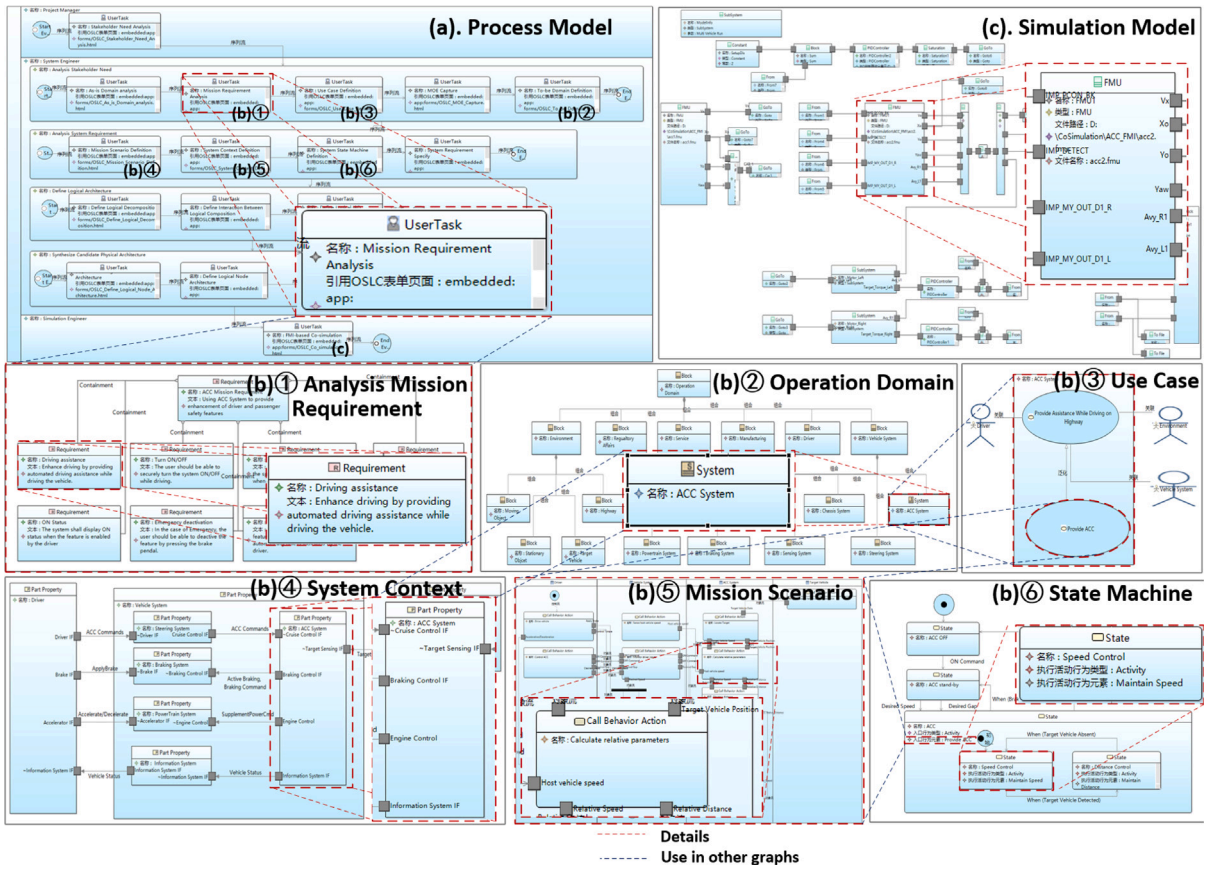

Fig. 7. DSM models in MetaGraph.

scenario, the system context of the ACC system is defined in Fig. 7(b)④. For example, the *ACC System* is developed as the instance of the meta-object *Part Property* with the instance of the meta-point *Braking Control IF*. Finally, the system statement machine is defined as shown in Fig. 7(b)⑥. For example, *Speed Control* is developed as the instance of the meta-object *State* with the instance of the meta-property *Entry Element*. The activity diagram *Provide ACC* is used in the state *ACC* as the start condition.

(3) Fig. 7(c) shows part of the instances for the simulation models. The model topology of the simulation models is used to build the Simulink models. For example, the instance of the meta-object *FMU* in the simulation model includes the instance of the meta-point, such as the interface to send the speed signal of the following vehicle *Vx*. The *FMU* corresponds to the block *Following Vehicle FMU* in the Simulink model, which is shown in Fig. 7(d).

Based on these DSM models, the Simulink models are developed to execute co-simulation based on FMI specification. The FMI specification defines a container and an interface named FMU to couple two or more simulation tools in a co-simulation environment. In this study, the FMUs are generated by the CarSim tool, which is used to set the parameters of a two-vehicle cruise scenario. The road parameters of the ACC scenarios are set as shown in Fig. 8(a) and include the road slope, curve length, etc. The vehicle parameters are set as in Figs. 8(b) and 8(c) and include cruise mode, initial velocity, etc. Then, based on the FMUs, the Simulink model of the ACC controller model is developed as shown in Fig. 8(d). The blocks, lines, subsystems, parameters, ports, and connectors in the Simulink model correspond to $Block_b^{(id,t)}$, $Line_c^{(id,t)}$, $Subsystem_d^{(id,t)}$, $Parameter_e^{(id,t)}$, $Port_f^{(id,t)}$, and $Connector_f^{(id,t)}$ in Eq. (4), respectively. The Simulink controller model includes the FMU blocks of the front and following vehicles, the radar sensor subsystem, the hydraulic pressure brake master cylinder (HPBMC) controller consisting of blocks and subsystems, and the left and right motor braking subsystems. The simulation results include the vehicle speeds of both vehicles, pressure of the HPBMC, braking torque of the left and right wheels on the following vehicle, and distance between the two vehicles, as shown in Fig. 8(d). The radar sensor subsystem is developed to monitor the distance and angle between the two vehicles, as shown in Fig. 8(e). The left and right motor braking subsystems are developed to control the braking torque of the left and right wheels on the following vehicle, as shown in Fig. 8(f).

Based on the mapping rules shown in Fig. 4, the OSLC services for stakeholder requirements, DSM models, and Simulink models are developed using several OSLC adapters that correspond to Eqs. (2), (3), and (4), respectively. For example, Fig. 9 shows the generated OSLC services of the Simulink models using the OSLC Simulink adapter. Based on Fig. 4 and Eq. (4), the Simulink model is considered a *ServiceProvider* concept, which provides the *Service* of query capability for the model topology, all subsystems, and all blocks in a Simulink model, as shown in Fig. 9(a). The model topology, subsystem, blocks, and their parameters are considered a *resource* concept. Fig. 9(b) shows the details of the Simulink model topology information, which can be found by clicking the 0th URL in the OSLC service list in Fig. 9(a). The *Service* will execute the API to query the model topology. By clicking the *execute co-simulation* and *get co-simulation result* buttons in the co-simulation control window, the simulation is automatically executed, and the results are displayed on the web page, such as the distance between two vehicles is 40.10 m at 20.11 s in Fig. 9(b). Fig. 9(c) shows the topology of the *Motor Left* subsystem in the Simulink model. By clicking the links of blocks Avy_L1 in the property window, the OSLC service for blocks Avy_L1 is shown as in Fig. 9(d), which includes

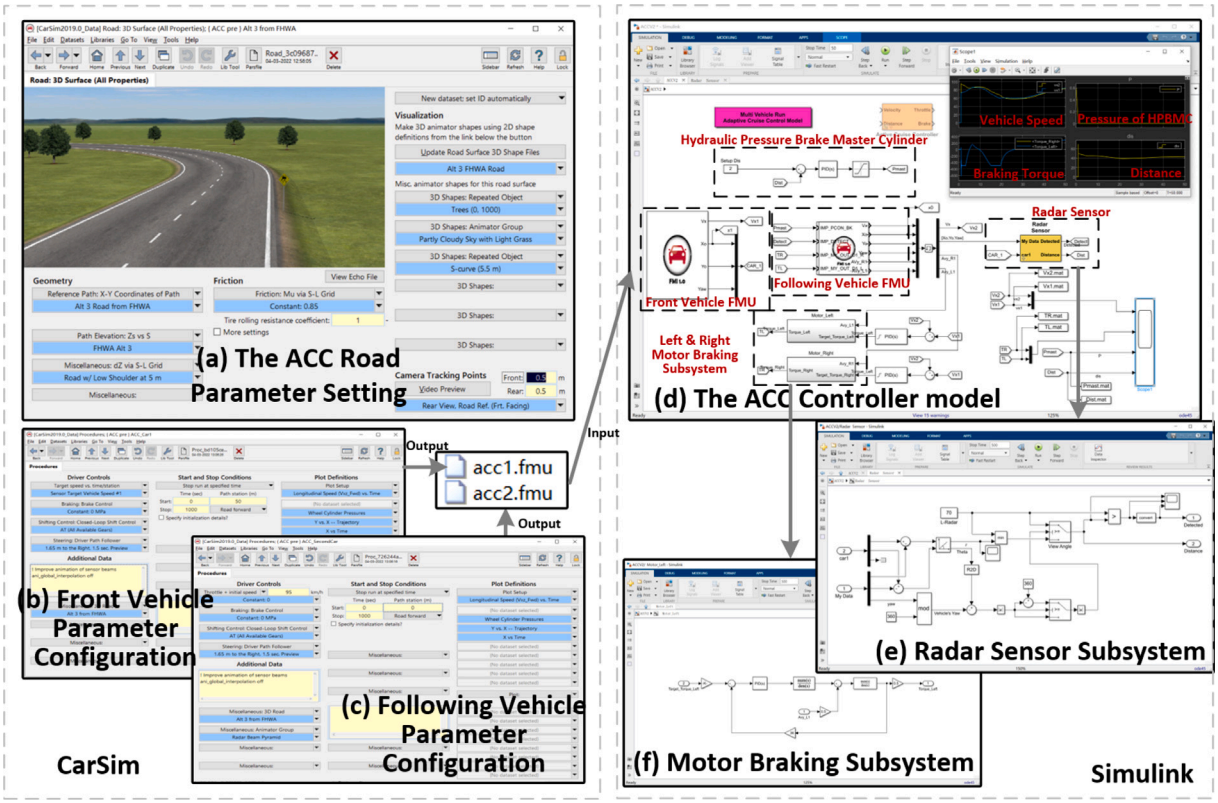

Fig. 8. Simulink model for co-simulation.

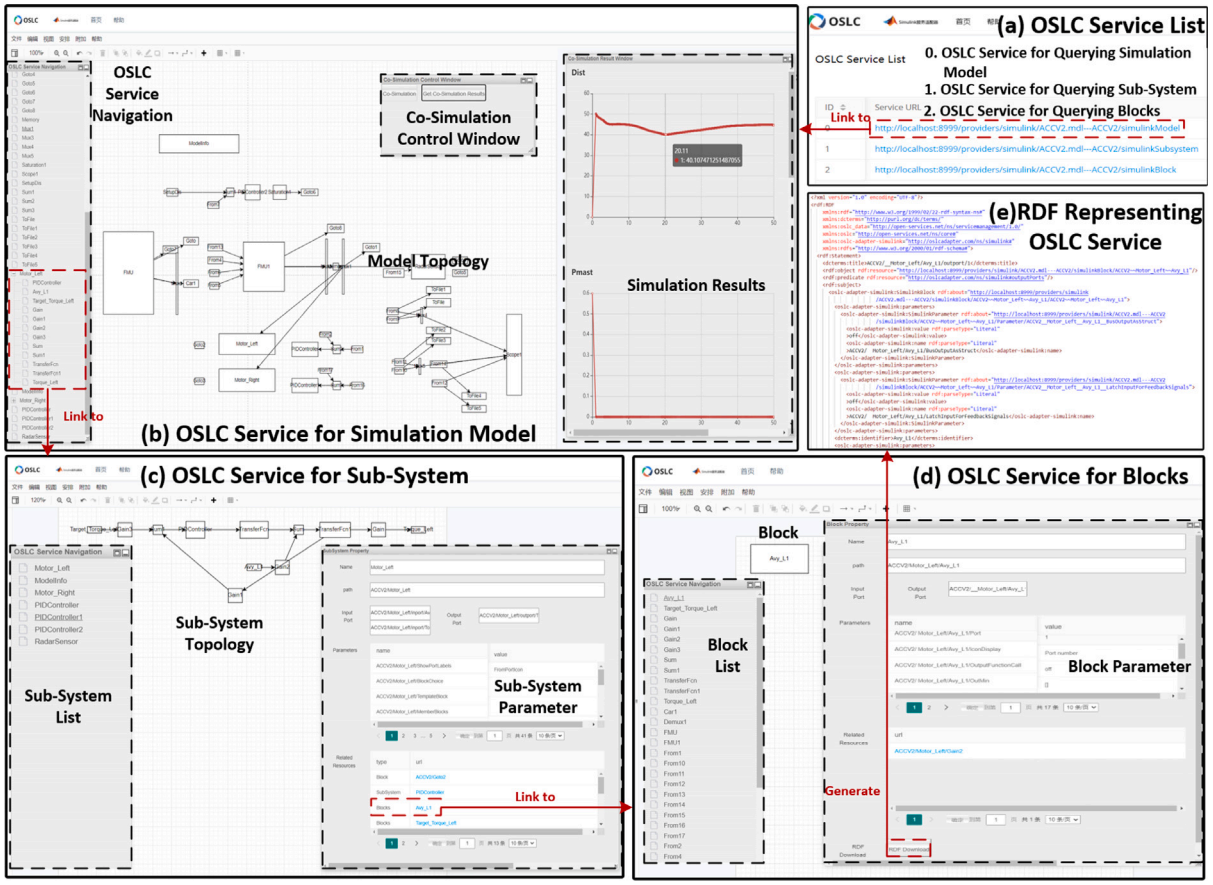

Fig. 9. OSLC service for co-simulation model in Simulink.

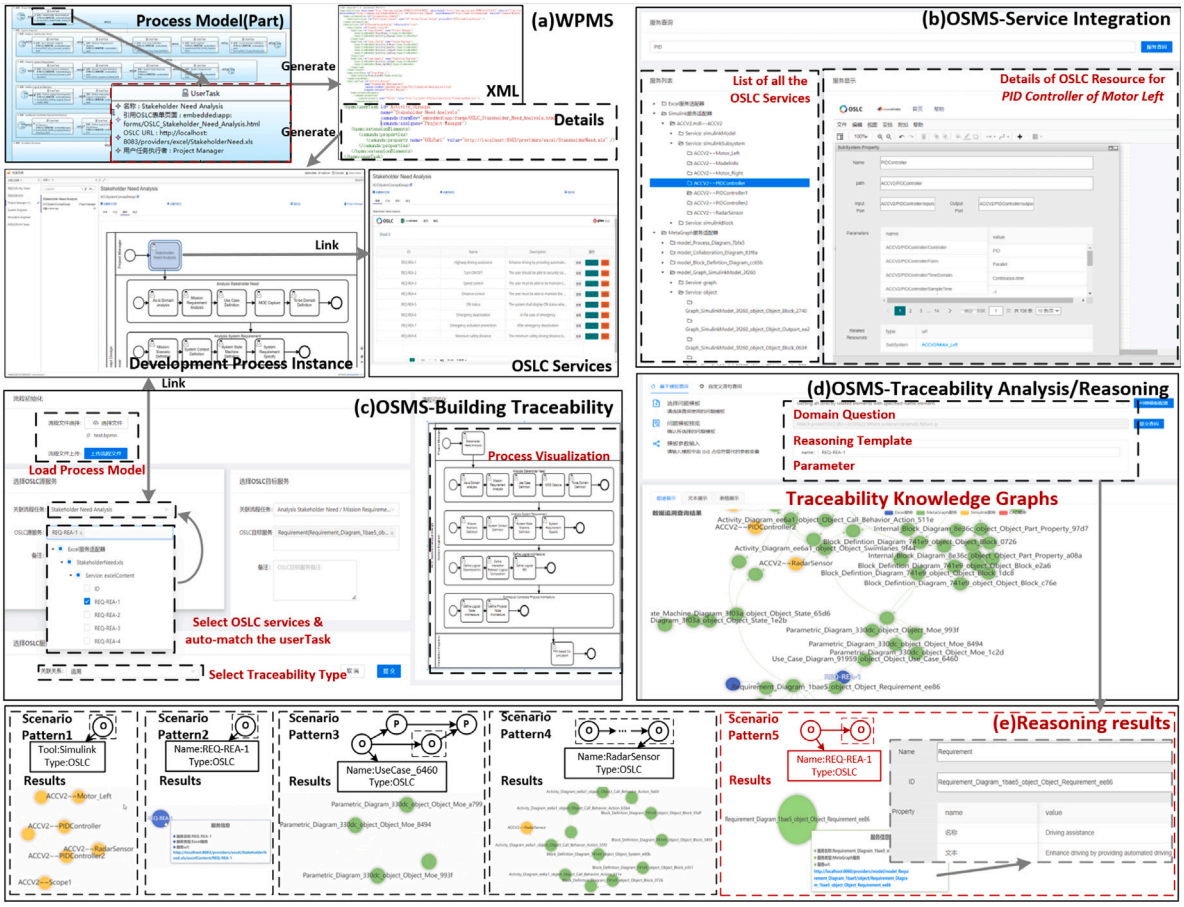

Fig. 10. OSMS supporting traceability management.

parameters such as port name and signal type. All the OSLC services generated are expressed in both HTML and RDF formats, as shown in Fig. 9(e).

5.3. OSMS supporting traceability construction and analysis

Fig. 10 shows the OSMS developed for traceability management. In Fig. 10(a), all the *UserTask* elements link with OSLC services through the corresponding URLs. Then, the process models are transformed into XML files. A compiler is developed to transform XML files into a WPMS linked with OSLC services, where the *Start*, *End*, *UserTask*, *Gateway*, *Sequence*, and *ServiceTask* in the WPMS correspond to the $Start_a$, End_b, $UserTask_c$, $Gateway_d$, $Sequence_e$, and $AutoTask_f$ in Eq. (6), respectively. In the WPMS, the *Scenario of managing traceability between stakeholders and work task* is realized. The work tasks are allocated to corresponding stakeholders to be executed, which corresponds to the $O\_Scenario_{PS}$ and $Allocation_i^{(allocationS,allocationT)}$ in Eq. (17). The stakeholders can manipulate the models for design through OSLC services binding to the work task of the WPMS. For example, the *Stakeholder needs definition* work task in the WPMS is linked to the related OSLC service of querying requirement items in Excel. In the WPMS, the requirement engineer manipulates OSLC services to modify stakeholder requirement in the Excel table, such as changing the *minimum safe driving distance* from 35 to 40 m.

In the service management module shown in Fig. 10(b), all the OSLC services are integrated into the OSMS to build traceability. Algorithm 1 shows the process of integrating OSLC services into the OSMS using a RESTful API, which corresponds to the concept O_Model in Eq. (1). In the *Procedure serviceIntegration*, all the root APIs of the OSLC adapters are provided. In the *Procedure Discovery*, a recursive algorithm is used to discover and traverse all the OSLC services until the OSLC *Resource* no longer links to another OSLC service URI. The stakeholders can access the OSLC service by searching the keywords. For example, if simulation engineers want to find the OSLC service related to the PID controller, they can search the keyword *PID* and click the label in the OSLC service list. The detailed information of the left motor *PIDController* is visualized in the OSMS, such as the proportional, integral, and derivative items of the controller.

In the OSMS, the *Scenario of managing traceability between OSLC services* and *Scenario of managing traceability between OSLC services and work tasks* discussed in Section 5.1 are realized by the traceability construction module. In Fig. 10(c), the traceability between OSLC services and the process is established in the traceability construction module. All OSLC services are listed as a tree structure in the OSMS. Algorithm 2 shows the process of automatic construction of the traceability between OSLC services and the process, which corresponds to the $O\_Scenario_{OP}$ and $ProcessLink_i^{(processLinkS,processLinkT)}$ in Eq. (11). In the *Procedure loadProcess*, all the elements in the XML file of the process model are captured. In the process model, the *UserTask* links to the OSLC service by adding the property of the OSLC service URI. To automatically match work tasks and OSLC services, the URI of the OSLC service is extracted as a particular criterion. In the *Procedure traceabilityConstruction*, the root URI of the selected OSLC service is extracted and compared to the URI attribute in the work tasks. If the two URIs match, the user task is automatically matched with the OSLC service. Subsequently, the traceability between OSLC services is established by combining the traceability relationships selected by the stakeholders, which corresponds to the $O\_Scenario_{OO}$ and $TraceLink_i^{(t,linkS,linkT)}$ in Eq. (9). For example, the engineer wants to established the traceability between the system requirement block *Adaptive Cruise Control* and the stakeholder's

Algorithm 1 Traversal algorithm for service integration

Input: OSLC services
Output: All the OSLC services integrated in the tree structure
Procedure serviceIntegration **begin**
serviceTree = ∅;
for *URL* from *OSLCServices* **do**
serviceTree = *serviceTree* ∪ *Discovery(URL)*
end for
return *serviceTree*;
Input: URL of OSLC services
Output: a set of discovery OSLC services
Procedure Discovery **begin**
serviceTree' = ∅;
if HttpRequest*(URL).result has ServiceProviderCatalogs* **then**
for *ServiceProviderCatalog* from *ServiceProviderCatalogs* **do**
Discovery(URL in ServiceProviderCatalog);
else if HttpRequest*(URL).result has ServiceProviders* **then**
for *ServiceProvider* from *ServiceProviders* **do**
Discovery(URL in ServiceProvider);
else if HttpRequest*(URL).result has Services* **then**
for *Service* from *Services* **do**
Discovery(URL in Service);
else if HttpRequest*(URL).result has Resources* **then**
serviceTree' = *serviceTree'* ∪ *Resources*;
return *serviceTree'*;
end if

need *REQ-REA-1* in the requirement table. First, the BPMN process model is imported and analyzed to capture all the work tasks in the development process. After selecting the requirement item *REQ-REA-1*, the corresponding work task *stakeholder need analysis*, the block *Adaptive Cruise Control*, and the work task *mission requirement analysis* are automatically selected to establish traceability. Then, by identifying the data interaction context in the development process, the relationship *trace to* between the OSLC services of *REQ-REA-1* and *Adaptive Cruise Control* is identified to establish traceability.

The *Scenario of analyzing traceability* discussed in Section 5.1 is realized by the traceability analysis module in the OSMS. In Fig. 10(d), the traceability between OSLC services and the process is analyzed with the traceability analysis module. For example, the stakeholder first selects the domain questions provided in the OSMS, such as *getting all directly related elements with specific-named element*, and fills in the parameter as a row ID of the requirement *REQ-REA-1*. The domain question corresponds to a reasoning template in the Cypher language [66]. The reasoning process is shown in Algorithm 3, which corresponds to the $TraceLink_i^{(t,linkS,linkT)}$ and $ProcessLink_i^{(processLinkS,processLinkT)}$ in Eqs. (9) and (11), respectively. The reasoning engine analyzes the structure of the reasoning template by matching the pre-defined patterns. The reasoning template above corresponds to the predefined *graph pattern*. Then, the reasoning engine executes the reasoning and returns the results (*p1*) in the corresponding format (*paths*). The reasoning results are shown in Fig. 10(e) and correspond to five given domain questions and scenario patterns, which are proposed in the *Scenario of analyzing traceability* discussed in Section 5.1 and illustrated as follows:

(1) *Scenario Pattern1* corresponds to the domain question ① *querying the single OSLC service based on the tool name for creating it*. The reasoning pattern is an OSLC service with the given tool name Simulink. The reasoning results show all OSLC services that have traceability and are created by Simulink that are found, such as the subsystems *ACCV2_Motor_Left* and *ACCV2_PIDController* in the Simulink models.
(2) *Scenario Pattern2* corresponds to the domain question ② *querying the single OSLC service based on its name*. The reasoning pattern is an OSLC service with a given name or ID *REQ-REA-1*. The reasoning results show the OSLC service of the Excel row for the stakeholder requirement *Highway Driving Assistance* in the requirement tables.
(3) *Scenario Pattern3* corresponds to the domain question ③ *querying any directly related OSLC services based on the sequence of corresponding work tasks*. The reasoning pattern consists of two related OSLC services and two related work tasks in the development process. One of the OSLC services has a given ID *UseCase_6460*, which corresponds to the use case *Provide Assistance While Driving on Highway*. The reasoning results show that four object instances of measure of effectiveness are designed based on this use case, such as *Cost* in the DSM models.
(4) *Scenario Pattern4* corresponds to the domain question ④ *querying any relevant OSLC services for the specific model element or data in*

Algorithm 2 Algorithm for traceability construction

Input: XML of Process model
Output: List of work tasks in Process model
Procedure loadProcess **begin**
XMLObject=loadXML(*XML*);
for *DOM* from *XMLObject* **do**
for *DOM.attribute* from *DOM* **do**
if *DOM is UserTask* **then**
if *DOM.attribute is OSLCURI* **then**
UserTaskObject.setAttribute(*DOM.attribute*.
getRootAddress());
else *UserTaskObject*.setAttribute(*DOM.attribute*);
UserTaskList.add(*UserTaskObject*);
end if
else
OtherObject.setAttribute(*DOM.attribute*);
OtherList.add(*OtherObject*);
end if
WorkTaskList.add(*UserTaskList*);
WorkTaskList.add(*OtherList*);
return *WorkTaskList*;
end for
end for
Input: All the OSLC services in the tree structure, XML of Process model
Output: Traceability relationships in knowledge graph
Procedure traceabilityConstruction **begin**
authority = *User*.sendUserInfoToAuthClient(*UserInfo*);
KG = *platform*.connectGraphDB(*authority*);
sourceService = User.selectSourceService(*serviceTree*);
targetService = User.selectTargetService(*serviceTree*);
relationship = User.selectRelationship();
WorkTaskList = loadProcess(*XML*);
for *service from [sourceService, targetService]* **do**
for *Task from WorkTaskList* **do**
ServiceURL = *service*.getRootAddress();
if *(Task is UserTask) and (Task.getAttribute(OSLCURI) is ServiceURL)* **then**
processTraceability = *KG*.buildTraceability(*service,'Belong to',Task*);
KG.checkandUpdateTraceability(*processTraceability*);
end if
traceability = *User*.buildTraceability
(*sourceService,relationship,targetService*);
KG.checkandUpdateTraceability(*traceability*);
end for
end for

the case study. The reasoning pattern specifies a chain, where two OSLC services are included as the source and target of this chain. One of the OSLC services has a given ID *ACCV2_RadarSensor*, which corresponds to the subsystem *RadarSensor* in the Simulink model. The reasoning results show that thirteen OSLC services are found that have potential relationships with *RadarSensor*, such as the object instance of *call action behavior* with the name *calculate relative speed* in the activity diagram *define mission scenarios*.

(5) *Scenario Pattern5* corresponds to the domain question ⑤ *querying any directly related OSLC services based on their names*. The reasoning pattern consists of two related OSLC services, one of which has a given ID of requirement *REQ-REA-1*. The reasoning results show that the requirement is related to the requirement block *Requirement_Diagram_1bae5_object_Object_Requirement_ee86* in the DSM model. The requirement block has a local label *Requirement* and property *text* with value *Enhance driving by providing automated driving assistance while driving the vehicle*.

Algorithm 3 Algorithm for traceability analysis for five scenarios Patterns

Input: Traceability relationships in knowledge graph,
Reasoning Templates (e.g: "Match p=(a:OSLC)-[b]->(c:OSLC) Return p")
Output: Analysis Result

```
Procedure traceabilityAnalysis begin
GraphTemplate ='(.*)=((.*))->[(.*)]->((.*))';
RelTemplate ='((.*))->[(.*)]->((.*))';
NodeTemplate ='((.*))';
PreDefineTemplateList.add(GraphTemplate,RelTemplate , NodeTemplate);
MatchPartInTemplate = Template.getPart('Match'); //e.g "p=(a:OSLC)-[b]->(c:OSLC)"
ReturnPartInTemplate = Template.getPart('Return'); //e.g "p"
for PreDefineTemplate from PreDefineTemplateList do
    TemplatePattern = Pattern.compile(PreDefineTemplate);
    Matcher = TemplatePattern.Matcher(MatchPartInTemplate);
    if Matcher.find() then
        if TemplatePattern is GraphTemplate then
            ElementList.add('Path': Matcher .group(1),
                            'Node':[ Matcher.group(2), Matcher.group(4)],
                            'Relationship': Matcher.group(3))
        else if TemplatePattern is RelTemplate then
            ElementList.add('Node':[ Matcher.group(1), Matcher.group(3)],
                            'Relationship': Matcher.group(2))
        else if TemplatePattern is NodeTemplate then
            ElementList.add('Node':[ Matcher.group(1)])
        end if
    end if
end for
Result = Neo4jDriver.run(Template);
if ElementList.findKey(ReturnPartInTemplate) is 'Path' then
    ReturnList.addPaths(Result);
else if ElementList.findKey(ReturnPartInTemplate) is 'Relationship' then
    ReturnList.addRelationships(Result);
else if ElementList.findKey(ReturnPartInTemplate) is 'Node' then
    ReturnList.addNodes(Result);
end if
return ReturnList;
```

5.4. Unified authentication system supporting authority management

The *Scenario of managing traceability between OSLC services and stakeholders* discussed in Section 5.1 is achieved by the unified authentication system in the MBSE toolchain. Fig. 11 shows the unified authentication system for realizing the authentication process, which corresponds to $Operation_i^{(ty,operationS,operationT)}$ in Eq. (14). As shown in Figs. 11(a) and 11(b), after entering the correct username and password, the list of stakeholders and their roles are provided based on the tables sys_user, sys_role, and sys_user_role in a MySQL database; the person with specific authority (e.g., administrator) can assign roles to stakeholders and configure their permissions, detailed in Fig. 11(c). In Fig. 11(d), the project refers to a specific Neo4j database in the docker container list. The process of project management is shown as follows: (1) The administrator sets up a project by specifying the project name and ID, as shown in Fig. 11(d)①. This ID is used not only for the Neo4j database in the docker container list but also for the project item in the MySQL database. (2) The current Neo4j database container is stopped, as shown in Fig. 11(d)②. (3) The user cannot find traceability of the previous project in the new project, as shown in Fig. 11(d)③. (4) The ports of the Neo4j database in the docker container list are automatically scheduled and mapped to a specific port, as shown in Fig. 11(d)④. Meanwhile, files of the Neo4j database are automatically created and mounted to the specified local path.

6. Discussion and evaluation

6.1. Qualitative evaluation

The integration of the MBSE tool-chain and approach into the industry provides the potential for increased automation and efficiency, and it builds an important foundation for IIIE. However, the issues about complex interrelationships and interoperability in traceability management scenarios between different domain tools are barriers that must be addressed. The work presented in this paper provides a new framework for designing MBSE tool-chain to promote the interoperability and provide semantic description for complex interrelationships in traceability management scenarios.

In the case study, the composition of the MBSE toolchain prototype is formalized using the CT ontology; thus, the completeness of the CT ontology is verified through the case study that implements the MBSE toolchain prototype. The five given scenarios in the case study are satisfied by the MBSE toolchain prototype, which means that the CT ontology supports the metric *logic in traceability management scenarios* given in Table 3. The detailed corresponding relationships between the CT ontology and MBSE toolchain in the case study are shown in Table 10. Through OSLC services, the WPMS enables the support of traceability management between tasks in the process and models. The OSMS enables the traceability construction, analysis, and visualization between models. The unified authentication system manages the traceability between models and stakeholders. Through the CT ontology, the topology interrelationships between the models, development processes, and stakeholders related to the MBSE toolchain are formalized, which promotes the efficiency of the MBSE toolchain development and semantic interoperability in traceability management scenarios.

In addition, the results of the assessment questions in Section 3.3 are used to qualitatively evaluate the metric *efficiency of traceability management* for the MBSE toolchain prototype, which is then compared with the traditional traceability management (TTM) approach using a design structure matrix. The evaluation results are listed in Table 11.

(1) *Promote the level of automation for the generation of trace links*: In TTM, the stakeholder uses a design structure matrix to establish traceability links. These links are often limited to a certain domain tool, which means generating traceability links between different domain tools can be difficult. In the MBSE toolchain

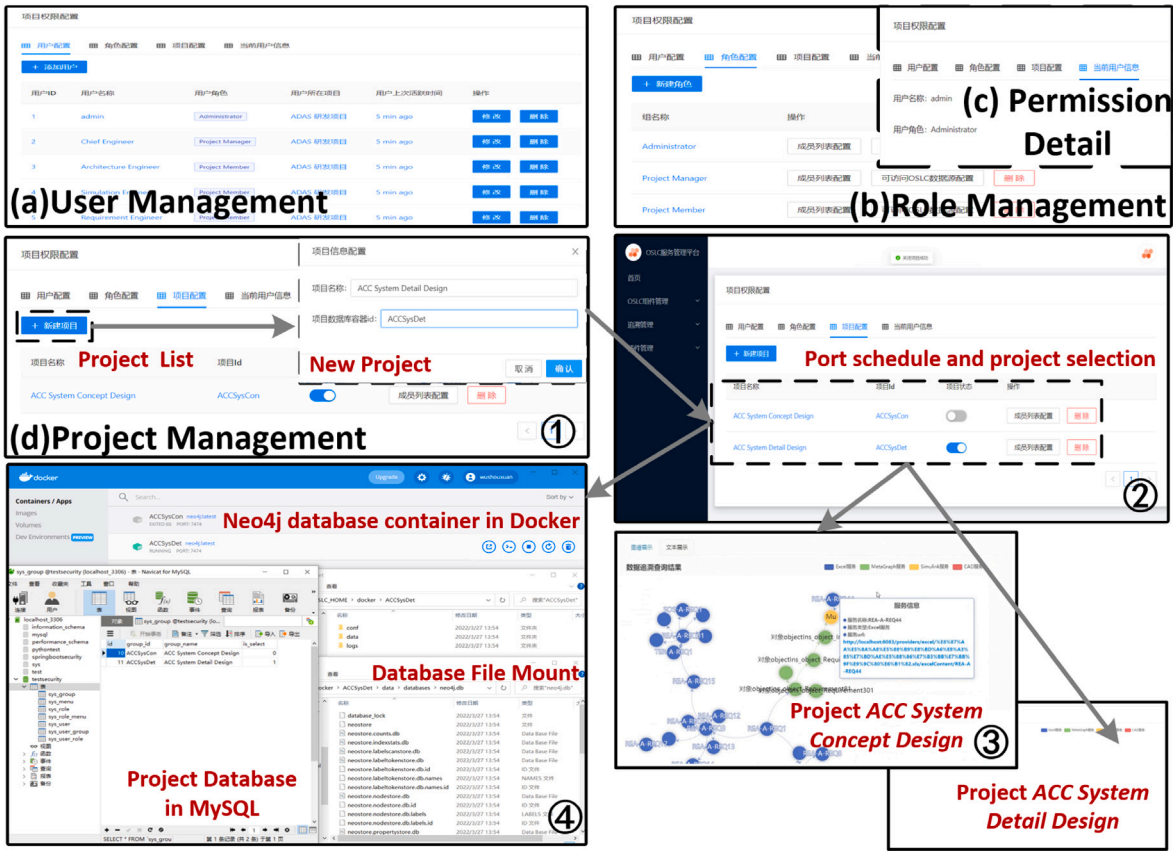

Fig. 11. Authority management process.

prototype, the OSLC adapters are used to generate OSLC services to construct traceability links. These OSLC services can be integrated quickly through a URI, as shown in Fig. 10(b).

(2) *Support storage of trace links*: In TTM, the storage of traceability links in the design structure matrix is often ignored or requires additional toolsets, such as a database for storage. In the MBSE toolchain prototype, the traceability links are stored in the Neo4j database in Docker containers, as shown in Fig. 11(d)③, which also enables the rapid deployment of the toolchain for distributed-development teams using images in Docker.

(3) *Support visualizing (representing) trace links*: In TTM, the stakeholder uses a design structure matrix to visualize traceability links. When the number of links is large enough, the visualization of the matrix has a negative impact on analyzing traceability. In the MBSE toolchain prototype, the traceability between OSLC services is displayed by table and graph views, as shown in Fig. 10. Moreover, OSLC services adopt the model perspectives that engineers are more accustomed to for visualization, which promotes the understanding of models for stakeholders.

(4) *Support creation, deletion, modification, and/or querying of trace links*: In TTM, the operation of traceability links in a design structure matrix requires additional tool sets. In the MBSE toolchain prototype, the OSMS is developed to support the creation, deletion, modification, and/or querying for traceability links by semi-automatic configuration, as shown in Fig. 10(c).

(5) *Promote interchange of traceability information*: In TTM, the stakeholders share the design structure matrix with each other through documents, e-mails, and meetings. In the MBSE toolchain prototype, the WPMS and traceability analysis plugin in Simulink are developed. The WPMS provides a unified platform to implement the development process and share information through manipulating models and data, as shown in Fig. 10(a). The OSLC services improve the information exchange between different domain tools, such as Excel, MetaGraph, and Simulink, as shown in Fig. 10(d). This information is used to support decision making in the design process, which promotes interchange of traceability information.

(6) *Analysis of the obtained traceability information*: In TTM, the traceability links in the design structure matrix need to be analyzed manually, or additional tool sets and algorithms are required for traceability analysis. In the MBSE toolchain prototype, the Cypher language is adopted in the OSMS to analyze the traceability between OSLC services and work tasks in the WPMS, as shown in Fig. 10(d). Through the domain questions, the reasoning templates are customized to analyze the topological relationships between models and work tasks to complete relationships between potential related elements.

6.2. Quantitative evaluation

Table 12 lists the interoperability metrics for the case study of the ACC system design. The interoperability metrics reflect the ability to exchange information between tools for the ACC system design. In the MBSE toolchain prototype, the information between different models is exchanged through OSLC services of models. Thus, the number of OSLC services indicates the interoperability between different models. In the case study, three OSLC adapters are developed for the ACC system design, one each for Excel, MetaGraph, and Simulink. There are a total of three URIs of the OSLC *ServiceProviderCatalog*, 66 URIs of the OSLC *ServiceProvider*, 109 URIs of the OSLC *Service*, and 261 URIs of the OSLC *Resource* that are generated from the OSLC adapters.

Table 10
Corresponding relationships between CT ontology and the MBSE toolchain.

Composition of the MBSE toolchain in the case study			CT Ontology
Models & Corresponding OSLC service	Stakeholder requirement tables in excel	Tables	$Table_b^{type}$
		Rows	Row_c^{type}
		Cells	$Cell_d^{type}$
	Meta-Models and instances of DSM models in MetaGraph	Meta-Graph of process	$Process_i$, $Graph_a^{(id,t)}$
		Meta-Object of start	$Start_a$, $Object_b^{(id,t)}$
		Meta-Object of end	End_b, $Object_b^{(id,t)}$
		Meta-Object of UserTask	$UserTask_c$, $Object_b^{(id,t)}$
		Meta-Object of ServiceTask	$AutoTask_f$, $Object_b^{(id,t)}$
		Meta-Object of gateway	$Gateway_d$, $Object_b^{(id,t)}$
		Meta-Relationship of sequence	$Sequence_e$, $Relationship_e^{(id,t)}$
		GOPPRR of system architecture model	$Graph_a^{(id,t)}$, $Object_b^{(id,t)}$, $Property_c^{(id,t)}$, $Point_d^{(id,t)}$, $Relationship_e^{(id,t)}$, $Role_f^{(id,t)}$
		GOPPRR of simulation model	$Graph_a^{(id,t)}$, $Object_b^{(id,t)}$, $Property_c^{(id,t)}$, $Point_d^{(id,t)}$, $Relationship_e^{(id,t)}$, $Role_f^{(id,t)}$
	Simulink models in simulink	Block	$Block_b^{(id,t)}$
		Line	$Line_c^{(id,t)}$
		Sub-System	$Subsystem_d^{(id,t)}$
		Parameter	$Parameter_e^{(id,t)}$
		Port	$Port_f^{(id,t)}$
		Connector	$Connector_g^{(id,t)}$
WPMS	Allocation of work tasks for stakeholders		$Allocation_i^{(allocationS,allocationT)}$
	Start		$Start_a$
	End		End_b
	UserTask		$UserTask_c$
			$ProcessLink_i^{(processLinkS,processLinkT)}$
	ServiceTask		$AutoTask_f$
	Gateway		$Gateway_d$
OSMS	Service management module		O_Model
	Traceability construction module		$TraceLink_i^{(t,linkS,linkT)}$
			$ProcessLink_i^{(processLinkS,processLinkT)}$
	Traceability analysis module		$TraceLink_i^{(t,linkS,linkT)}$
			$ProcessLink_i^{(processLinkS,processLinkT)}$
Unified authentication system	User management		$Stakeholder_b$, API_e^t, $Operation_i^{(ty,operationS,operationT)}$
	Role management		$Role_c$, API_e^t, $Operation_i^{(ty,operationS,operationT)}$
	Permission management		$Permission_d$, API_g^t, $Operation_i^{(ty,operationS,operationT)}$
	Project management		$Project_a$, $Stakeholder_b$, API_e^t, $Operation_i^{(ty,operationS,operationT)}$
	Authentication process		$Operation_i^{(ty,operationS,operationT)}$

Table 11
The evaluation results for assessment problem in the case study.

Question	Score
Level of automation for the generation of trace links	Y
Storage of trace links	Y
Mechanism for visualizing (representing) trace links	Y
Creation, deletion, modification and/or querying of trace links	Y
Interchange of traceability information	Y
Analysis with the traceability information obtained	Y

Compared with that in TTM, these OSLC services significantly promote the interoperability between models in Excel, MetaGraph 2.0, and Simulink. The CT ontology formalizes the elements included in the OSLC services. Therefore, we infer that the CT ontology can also promote interoperability of the models.

In the case study, the capability for cognitive reasoning to analyze traceability is measured by the number of traceability links that are established in the knowledge graphs and total number of reasoning templates and reasoning results. The capability for cognitive reasoning reflects the ability of integrating and analyzing the interrelationships between information. There are 110 traceability links between OSLC services and 56 traceability links between OSLC services and work tasks in the Neo4j database, as seen in Fig. 10(d). The process of establishing this traceability corresponds to the process of forming knowledge. Additionally, five domain problems are provided to analyze the traceability in *Goal for analyzing traceability* in Section 5.1. In the case study, there are 26 OSLC services found in the reasoning results, as shown in Fig. 10(d). The process of generating these reasoning results corresponds to the process of applying knowledge. The CT ontology is developed to formalize the elements related to the traceability analysis process. Therefore, we infer that the CT ontology has the capability of cognitive reasoning to analyze traceability, which can be further used to identify the potential topology and semantic connections between the model and process.

In the case study, the levels of communication between stakeholders in the development processes are assessed by the number of stakeholders, organizations, and work tasks in the MBSE toolchain. The levels of communication reflects the ability to exchange multi-discipline information among stakeholders through information and communication technology. There are five users in the unified authentication system, including the administrator, project manager, and several engineers. These users are involved in two projects: the ACC system concept design and ACC system detail design. The users are divided into three roles with different permissions. For example, the user *admin* has the role *administrator* to manage user and role lists, as shown in Figs. 11(a)(b)(c). Additionally, there are 16 work tasks in the

Table 12
The metrics in the case study of ACC system design.

Measurement	Metric	Counts
Interoperability of models	Number of OSLC adapter	3
	Number of OSLC ServiceProviderCatalog	3
	Number of OSLC ServiceProvider	66
	Number of OSLC Service	109
	Number of OSLC Resource	261
Capability for reasoning to analyze traceability	Number of domain question query templates types	5
	Total Number of reasoning result for domain questions	26
	Number of traceability links between OSLC service	110
	Number of traceability links between OSLC service and work tasks	56
Levels of Communication between Stakeholders in development Processes	Number of users in OSMS	5
	Number of roles in OSMS	3
	Number of permissions in OSMS	16
	Number of projects in OSMS	2
	Number of work tasks in process models	11

WPMS, as shown in Figs. 10(a) and 7(a). The unified authentication system ensures the OSLC services can be accessed by the users with the correct permissions. The WPMS helps engineers implement their work in a unified platform and workflow, which promotes the process management capabilities for the co-design process. The CT ontology is developed to formalize the elements related to the unified authentication system and WPMS. Therefore, we infer that the CT ontology can promote the levels of communication between stakeholders.

6.3. Scalability discussion

Internal validity and external validity are two important aspects of a case study approach [73]. Therefore, internal and external validity are used to evaluate the contribution of this paper. In this paper, the internal validity of the case study refers to **the increased efficiency due to the use of the proposed tool-chain and the CT ontology in traceability management scenarios**, which is evaluated by the qualitative approach and the quantitative evaluation approach mentioned in Sections 6.1 and 6.2. The external validity of the case study refers to **the extent to which the proposed toolchain and the CT ontology can be replicated in other traceability management scenarios**, representing the scalability of this case study. The external validity of the case study are discussed from the perspectives of the toolchain and the ontology design, respectively.

From the perspective of the toolchain, although the design methods, design processes, and design objectives of complex systems in the aerospace domain may differ from those of the adaptive cruise control system, the design tools used are similar. According to industry practice, two of the most commonly used MBSE modeling and verification tools in Chinese industry are the SysML-based tools and matlab-language-based tools [74]. In the proposed toolchain prototype, the adopted modeling tool MetaGraph can be used to develop DSM models. These DSM models can be used to represent model information that required by SysML and other modeling languages, and they can be further transformed to proprietary formats required in different modeling tools (such as .m script in Matlab). Thus, we can infer that the traceability management scenarios of the case study in this paper can be replicated in other traceability management scenarios in the aerospace domain.

From the perspective of the ontology design, the design purpose of the CT ontology is to formalize the concepts of models, development processes, stakeholders, and their interrelationships in traceability management scenarios. The development process of aerospace products is also composed of specific design tasks and assigned to the corresponding stakeholders (such as requirements managers, engineers in various domains) for collaborative design. A large number of descriptive models (such as diagrams in SysML models) and analytical models (such as Simulink and Modelica models) are developed and transferred between different design tasks to support the consistent transmission of design information. Similar to the traceability management of models for ACC system design, the traceability management of models for product design in aerospace and other fields can also be characterized by the concepts of development process, stakeholders, related concepts of the models, and their interrelationships by the designed CT ontology. This ontology provides a domain-independent representation, facilitating a unified representation of related concepts in traceability management scenarios, thereby ensuring that all stakeholders involved in the design process share a common understanding of these scenarios.

6.4. Limitations

The case study of an ACC system is adopted to evaluate the proposed CT ontology. However, this study only consists of a two-vehicle cruise, without considering the actual road conditions, such as vehicles changing lanes. This study has several limitations. (1) The impact of dynamic changes in the work tasks on the traceability management scenarios are not considered. Thus, a dynamic choreography method to support reconfiguration of work tasks deployed in the WPMS will be developed in the future. (2) The designed CT ontology is limited to the MBSE toolchain developed in this study. Other scenarios for traceability management are not included, such as scenarios integrating information of CAD models or internet of things (IoT) data. These scenarios will be considered in the future. (3) The completeness of the CT ontology is verified by a case study of an ACC system design in this study. Formal verification of the functionalities of the CT ontology will be developed using OWL in the future. (4) Although the concepts and relationships of proposed CT ontology are identified based on the developed toolchain prototype, there are alternative and possible technical solutions that can be used to implement the CT ontology for future research. Due to the strong semantic compatibility with domain-specific modeling languages for the KARMA language, one promising avenue could involve exploring the potential integration of domain-specific modeling languages, such as SysML, UPDM, or EAST-ADL, with the CT ontology. In addition, OSLC services of models provide tool integration ability for traceability management scenarios in this toolchain. Another possible alternative solution of OSLC is to investigate the adoption of emerging semantic web technologies, such as OWL (Web Ontology Language) together with service-oriented architecture. It can provide more robust mechanisms for representing and reasoning about complex semantic relationships between heterogeneous models (such as subclass and equivalence relationships).

7. Conclusion

For facilitating the industrial information integration process to support IIIE, the traceability management in MBSE implementation should be first considered. In this paper, we propose a CT ontology for traceability management in an MBSE toolchain that is developed based

on an MBSE toolchain prototype and formalizes models, stakeholders, and development processes, as well as traceability management scenarios. Through a case study of an ACC system, the completeness of the CT ontology is evaluated by qualitative and quantitative analyses of the proposed MBSE toolchain. The results demonstrate that the CT ontology helps stakeholders manage and analyze traceability between heterogeneous models in the development process. To improve the efficiency of information integration, more discipline information can be considered and integrated into the proposed CT ontology in the future. Besides, we intend to extend the ontology of the CT to support other complex scenarios during MBSE implementation, such as automatic change management and consistency management for MBSE models.

CRediT authorship contribution statement

Shouxuan Wu: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Guoxin Wang:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing. **Jinzhi Lu:** Conceptualization, Formal analysis, Project administration, Resources, Software, Supervision, Writing – review & editing. **Zhenchao Hu:** Software. **Yan Yan:** Funding acquisition, Project administration, Resources. **Dimitris Kiritsis:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgments

This work was supported by the national key research and development program of China (Grant No. 2020YFB1708100), the CN ministry project (Grant No. 50923010101), and the “111 Center” B18002.

References

[1] L. Da Xu, Enterprise systems: State-of-the-art and future trends, IEEE Trans. Ind. Inform. 7 (4) (2011) 630–640.
[2] L.D. Xu, Industrial information integration — An emerging subject in industrialization and informatization process, J. Ind. Inf. Integr. 17 (2020) 100128, http://dx.doi.org/10.1016/j.jii.2020.100128.
[3] Y. Chen, Industrial information integration — A literature review 2006–2015, J. Ind. Inf. Integr. 2 (2016) 30–64, http://dx.doi.org/10.1016/j.jii.2016.04.004.
[4] Y. Chen, A survey on industrial information integration 2016–2019, J. Ind. Integr. Manag. 5 (01) (2020) 33–163.
[5] N. Li, L. Zhao, C. Bao, G. Gong, X. Song, C. Tian, A real-time information integration framework for multidisciplinary coupling of complex aircrafts: An application of IIIE, J. Ind. Inf. Integr. 22 (2021) 100203, http://dx.doi.org/10.1016/j.jii.2021.100203.
[6] B. Wang, Y. Zhang, G. Su, An integrated approach for electromagnetic compatible commercial aircraft engine cable harnessing, J. Ind. Inf. Integr. 27 (2022) 100344, http://dx.doi.org/10.1016/j.jii.2022.100344.
[7] B.A. Mousavi, C. Heavey, R. Azzouz, H. Ehm, C. Millauer, R. Knobloch, Use of model-based system engineering methodology and tools for disruption analysis of supply chains: A case in semiconductor manufacturing, J. Ind. Inf. Integr. 28 (2022) 100335, http://dx.doi.org/10.1016/j.jii.2022.100335.
[8] L. Da Xu, Enterprise Integration and Information Architecture: A Systems Perspective on Industrial Information Integration, CRC Press, 2014.
[9] T.D. West, A. Pyster, Untangling the digital thread: The challenge and promise of model-based engineering in defense acquisition, Insight 18 (2) (2015) 45–55.
[10] V. Singh, K.E. Willcox, Engineering design with digital thread, AIAA J. 56 (11) (2018) 4515–4528.
[11] J.A. Estefan, et al., Survey of model-based systems engineering (MBSE) methodologies, Incose MBSE Focus Group 25 (8) (2007) 1–12.
[12] Y. Chen, J. Jupp, Model-based systems engineering and through-life information management in complex construction, in: IFIP International Conference on Product Lifecycle Management, Springer, 2018, pp. 80–92.
[13] A. Mengist, L. Buffoni, A. Pop, An integrated framework for traceability and impact analysis in requirements verification of cyber–physical systems, Electronics 10 (8) (2021) 983.
[14] A. Espinoza, G. Botterweck, J. Garbajosa, A formal approach to reuse successful traceability practices in SPL projects, in: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 2352–2359.
[15] A.M. Madni, M. Sievers, Model-based systems engineering: Motivation, current status, and research opportunities, Syst. Eng. 21 (3) (2018) 172–190.
[16] J. Fitzgerald, C. Gamble, P.G. Larsen, K. Pierce, J. Woodcock, Cyber-physical systems design: Formal foundations, methods and integrated tool chains, in: 2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering, IEEE, 2015, pp. 40–46.
[17] T. Karhela, A. Villberg, H. Niemistö, Open ontology-based integration platform for modeling and simulation in engineering, Int. J. Model. Simul. Sci. Comput. 3 (02) (2012) 1250004.
[18] A.M. Madni, D. Erwin, C.C. Madni, Digital twin-enabled MBSE testbed for prototyping and evaluating aerospace systems: Lessons learned, in: 2021 IEEE Aerospace Conference, 50100, IEEE, 2021, pp. 1–8.
[19] S. Wu, J. Lu, Z. Hu, P. Yang, G. Wang, D. Kiritsis, Cognitive thread supports system of systems for complex system development, in: 2021 16th International Conference of System of Systems Engineering, SoSE, IEEE, 2021, pp. 82–87.
[20] J. Lu, J. Wang, D. Chen, J. Wang, M. Törngren, A service-oriented tool-chain for model-based systems engineering of aero-engines, IEEE Access 6 (2018) 50443–50458.
[21] E. Kraft, HPCMP create™-AV and the air force digital thread, in: 53rd AIAA Aerospace Sciences Meeting, 2015, pp. 1–13, http://dx.doi.org/10.2514/6.2015-0042.
[22] D. Johnson, S. Speicher, Open services for lifecycle collaboration-core specification version 2.0, 2013, http://open-services.net/.
[23] I. Javid, “Evaluating ARCADIA using the FEMMP framework” (evaluating and comparing MBSE methodologies) university of applied sciences ingolstadt department of engineering and management (intern. industrial engineering) bachelor of engineering (b. eng.), 20, 2020, pp. 0–77, http://dx.doi.org/10.13140/RG.2.2.27012.55680.
[24] T. Blockwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, et al., Functional mockup interface 2.0: The standard for tool independent exchange of simulation models, in: Proc. 9th Int. Model. Conf. Sept. 3–5, 2012, Munich, Ger, vol. 76, 2012, pp. 173–184, http://dx.doi.org/10.3384/ecp12076173.
[25] J. Lu, J. Ma, X. Zheng, G. Wang, H. Li, D. Kiritsis, Design ontology supporting model-based systems engineering formalisms, IEEE Syst. J. 14 (8) (2021) 1–12, http://dx.doi.org/10.1109/JSYST.2021.3106195.
[26] H. Lykins, S. Friedenthal, A. Meilich, 4.4.4 Adapting UML for an object oriented systems engineering method (OOSEM), in: INCOSE International Symposium, vol. 10, (1) Wiley Online Library, 2000, pp. 490–497.
[27] D. Ferraiolo, J. Cugini, D.R. Kuhn, et al., Role-based access control (RBAC): Features and motivations, in: Proceedings of 11th Annual Computer Security Application Conference, 1995, pp. 241–248.
[28] R. Mannadiar, H. Vangheluwe, Domain-specific engineering of domain-specific languages, in: Proceedings of the 10th Workshop on Domain-Specific Modeling, DSM ’10, Association for Computing Machinery, New York, NY, USA, 2010, http://dx.doi.org/10.1145/2060329.2060356.
[29] M. von Rosing, S. White, F. Cummins, H. de Man, Business process model and notation-BPMN, 2015, pp. 429–453, http://dx.doi.org/10.1016/B978-0-12-799959-3.00021-5.
[30] IEEE, IEEE standard glossary of software engineering terminology, IEEE Std 610.12-1990, 1990, pp. 1–84, http://dx.doi.org/10.1109/IEEESTD.1990.101064.
[31] A. Corallo, M.E. Latino, M. Menegoli, P. Pontrandolfo, A systematic literature review to explore traceability and lifecycle relationship, Int. J. Prod. Res. 58 (15) (2020) 4789–4807.
[32] I.C. on Systems Engineering (Ed.), INCOSE Systems Engineering Handbook, vol. 2.0, 2000.
[33] S.F. Königs, G. Beier, A. Figge, R. Stark, Traceability in systems engineering–review of industrial practices, state-of-the-art technologies and new research solutions, Adv. Eng. Inform. 26 (4) (2012) 924–940.
[34] G. Fei, J. Gao, D. Owodunni, X. Tang, A model-driven and knowledge-based methodology for engineering design change management, Comput.-Aided Des. Appl. 8 (3) (2011) 373–382.
[35] J. Tao, S. Yu, A meta-model based approach for LCA-oriented product data management, Procedia CIRP 69 (2018) 423–428.
[36] S.D. Eppinger, T.R. Browning, Design Structure Matrix Methods and Applications, MIT Press, 2012.
[37] W.-T. Lee, W.-Y. Deng, J. Lee, S.-J. Lee, Change impact analysis with a goal-driven traceability-based approach, Int. J. Intell. Syst. 25 (8) (2010) 878–908, http://dx.doi.org/10.1002/int.20443.

[38] A. Sharon, D. Dori, O. De Weck, Model-based design structure matrix: Deriving a dsm from an object-process model, in: Second International Symposium on Engineering Systems, 2009, pp. 1–12.
[39] R. Brahmi, M. Hammadi, J.-Y. Choley, M. Trigui, N. Aifaoui, A SysML profile for mechanical assembly, in: 2020 IEEE International Systems Conference, SysCon, IEEE, 2020, pp. 1–7.
[40] D. Tang, R. Zhu, J. Tang, R. Xu, R. He, Product design knowledge management based on design structure matrix, Adv. Eng. Inform. 24 (2) (2010) 159–166.
[41] J. Guo, G. Wang, J. Lu, J. Ma, M. Törngren, General modeling language supporting model transformations of MBSE (part 2), in: INCOSE International Symposium, vol. 30, (1) Wiley Online Library, 2020, pp. 1460–1473.
[42] M. Kharrat, O. Penas, R. Plateaux, J.-Y. Choley, H. Trabelsi, J. Louati, M. Haddar, Integration of electromagnetic constraints as of the conceptual design through an MBSE approach, IEEE Syst. J. 15 (1) (2020) 747–758.
[43] J. Ma, G. Wang, J. Lu, H. Vangheluwe, D. Kiritsis, Y. Yan, Systematic literature review of MBSE tool-chains, Appl. Sci. 12 (7) (2022) http://dx.doi.org/10.3390/app12073431.
[44] J. Lu, D. Chen, G. Wang, D. Kiritsis, M. Törngren, Model-based systems engineering tool-chain for automated parameter value selection, IEEE Trans. Syst. Man Cybern. Syst. 52 (4) (2022) 2333–2347, http://dx.doi.org/10.1109/TSMC.2020.3048821.
[45] J. Lu, D.-J. Chen, D. Gürdür, M. Törngren, An investigation of functionalities of future tool-chain for aerospace industry, in: INCOSE International Symposium, vol. 27, (1) Wiley Online Library, 2017, pp. 1408–1422.
[46] D. Gürdür, J. El-Khoury, T. Seceleanu, L. Lednicki, Making interoperability visible: Data visualization of cyber-physical systems development tool chains, J. Ind. Inf. Integr. 4 (2016) 26–34, http://dx.doi.org/10.1016/j.jii.2016.09.002.
[47] J. El-khoury, A. Berezovskyi, M. Nyberg, An industrial evaluation of data access techniques for the interoperability of engineering software tools, J. Ind. Inf. Integr. 15 (2019) 58–68, http://dx.doi.org/10.1016/j.jii.2019.04.004.
[48] P.G. Larsen, J. Fitzgerald, J. Woodcock, P. Fritzson, J. Brauer, C. Kleijn, T. Lecomte, M. Pfeil, O. Green, S. Basagiannis, et al., Integrated tool chain for model-based design of cyber-physical systems: The INTO-CPS project, in: 2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS, CPS Data, IEEE, 2016, pp. 1–6.
[49] T.A. Bapty, J. Scott, S. Neema, R. Owens, Integrated modeling and simulation for cyberphysical systems extending multi-domain M&S to the design community, in: Proceedings of the Symposium on Model-Driven Approaches for Simulation Engineering, 2017, pp. 1–12.
[50] B. Chandrasekaran, J.R. Josephson, V.R. Benjamins, What are ontologies, and why do we need them? IEEE Intell. Syst. Appl. 14 (1) (1999) 20–26, http://dx.doi.org/10.1109/5254.747902.
[51] J. Lu, G. Wang, M. Törngren, Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain, IEEE Syst. J. 14 (1) (2019) 1297–1308.
[52] C. Hennig, A. Viehl, B. Kämpgen, H. Eisenmann, Ontology-based design of space systems, in: International Semantic Web Conference, Springer, 2016, pp. 308–324.
[53] R. Chen, Y. Liu, X. Ye, Ontology based behavior verification for complex systems, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 51739, American Society of Mechanical Engineers, pp. 1–9, V01BT02A038.
[54] N. Narayan, B. Bruegge, A. Delater, B. Paech, Enhanced traceability in model-based CASE tools using ontologies and information retrieval, in: 2011 4th International Workshop on Managing Requirements Knowledge, IEEE, 2011, pp. 24–28.
[55] P. Avgeriou, J. Grundy, J.G. Hall, P. Lago, I. Mistrík, Relating Software Requirements and Architectures, Springer Science & Business Media, 2011.
[56] J.H. Panchal, C.J.J. Paredis, J.K. Allen, F. Mistree, Managing design process complexity: A value-of-information based approach for scale and decision decoupling, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 48078, 2007, pp. 633–647.
[57] R. Wang, G. Wang, Y. Yan, M. Sabeghi, Z. Ming, J.K. Allen, F. Mistree, Ontology-based representation of meta-design in designing decision workflows, J. Comput. Inf. Sci. Eng. 19 (1) (2019).
[58] D.A. Kinard, F-35 digital thread and advanced manufacturing, in: 2018 Aviat. Technol. Integr. Oper. Conf, From Concept to Cockpit, The F-35 Lightning II, 2018, pp. 1–19, http://dx.doi.org/10.2514/6.2018-3369.
[59] M.A. Bone, M.R. Blackburn, D.H. Rhodes, D.N. Cohen, J.A. Guerrero, Transforming systems engineering through digital engineering, J. Def. Model. Simul. 16 (4) (2019) 339–355.
[60] T.D. West, M. Blackburn, Is digital thread/digital twin affordable? A systemic assessment of the cost of DoD's latest manhattan project, Procedia Comput. Sci. 114 (2017) 47–56.
[61] M. Bone, M. Blackburn, B. Kruse, J. Dzielski, T. Hagedorn, I. Grosse, Toward an interoperability and integration framework to enable digital thread, Systems 6 (4) (2018) 46, http://dx.doi.org/10.3390/systems6040046.
[62] T.D. Hedberg Jr., M. Bajaj, J.A. Camelio, Using graphs to link data across the product lifecycle for enabling smart manufacturing digital threads, J. Comput. Inf. Sci. Eng. 20 (1) (2020) 011011.
[63] S. Kwon, L.V. Monnier, R. Barbau, W.Z. Bernstein, Enriching standards-based digital thread by fusing as-designed and as-inspected data using knowledge graphs, Adv. Eng. Inform. 46 (2020) 101102.
[64] M. Helu, A. Joseph, T. Hedberg Jr., A standards-based approach for linking as-planned to as-fabricated product data, CIRP Ann 67 (1) (2018) 487–490.
[65] H.W. Lawson, A Journey Through the Systems Landscape, College Publications, 2010.
[66] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1433–1445.
[67] I. Santiago, A. Jiménez, J.M. Vara, V. De Castro, V.A. Bollati, E. Marcos, Model-driven engineering as a new landscape for traceability management: A systematic literature review, Inf. Softw. Technol. 54 (12) (2012) 1340–1356, http://dx.doi.org/10.1016/j.infsof.2012.07.008.
[68] J. Lu, D. Chen, J. Wang, M. Torngren, Towards a service-oriented framework for MBSE tool-chain development, in: 2018 13th Annual Conference on System of Systems Engineering, SoSE, IEEE, 2018, pp. 568–575, http://dx.doi.org/10.1109/SYSOSE.2018.8428746.
[69] S. Friedenthal, D. Dori, Y. Mordecai, Anupama, Types of models, in SEBoK v.2.5, 2021, available at: https://www.sebokwiki.org/wiki/Types_of_Models.
[70] J.M. Alvarez-Rodríguez, R.M. Zuñiga, J. Llorens, Elevating the meaning of data and operations within the development lifecycle through an interoperable toolchain, in: INCOSE International Symposium, vol. 29, (1) Wiley Online Library, 2019, pp. 1053–1071.
[71] D.L. McGuinness, F.v. Harmelen, OWL web ontology language overview, http://www.w3.org/TR/owl-features/.
[72] Object Management Group (OMG), OMG systems modeling language (OMG SysML™), 2012.
[73] N. Salkind, Internal and external validity, SAGE Dict. Quant. Manag. Res. (2011) 148–149.
[74] J. Lu, Y. Wen, Q. Liu, D. Gürdür, M. Törngren, MBSE applicability analysis in Chinese industry, in: INCOSE International Symposium, vol. 28, (1) Wiley Online Library, 2018, pp. 1037–1051.