The International Journal of Advanced Manufacturing Technology (2022) 121:5835–5854
https://doi.org/10.1007/s00170-022-09610-5

ORIGINAL ARTICLE

Exploring the concept of Cognitive Digital Twin from model-based systems engineering perspective

Lu Jinzhi[1] · Yang Zhaorui[2] · Zheng Xiaochen[1] · Wang Jian[2] · Kiritsis Dimitris[2]

Received: 9 March 2022 / Accepted: 23 June 2022 / Published online: 29 July 2022

Abstract

Digital Twin technology has been widely applied in various industry domains. Modern industrial systems are highly complex consisting of multiple interrelated systems, subsystems and components. During the lifecycle of an industrial system, multiple digital twin models might be created related to different domains and lifecycle phases. The integration of these relevant models is crucial for creating higher-level intelligent systems. The Cognitive Digital Twin (CDT) concept has been proposed to address this challenge by empowering digital twins with augmented semantic capabilities. It aims at identifying the dynamics and interrelationships of virtual models, thus to enhance complexity management capability and to support decision-making during the entire system lifecycle. This paper aims to explore the CDT concept and its core elements following a systems engineering approach. A conceptual architecture is designed according to the ISO 42010 standard to support CDT development; and an application framework enabled by knowledge graph is provided to guide the CDT applications. In addition, an enabling tool-chain is proposed corresponding to the framework to facilitate the implementation of CDT. Finally, a case study is conducted, based on simulation experiments as a proof-of-concept.

Keywords Cognitive Digital Twin · Digital Twin · Knowledge graph · Semantic modelling · Model-based systems engineering · KARMA language

1 Introduction

The complexity of modern industrial systems is continuously increasing. A highly complex industrial system, such as a smart manufacturing system, can be defined as a system-of-systems (SoS) [1]. SoS is a large-scale integrated system with multiple independent systems working collectively for a common mission [2]. Each of the systems may consist of many interconnected subsystems and components. Empowered by Information and Communication Technologies (ICT) and Cyber-physical Systems (CPS), a large number of virtual entities, such as data, information and knowledge related to the systems, subsystems and components, are generated, which compose the virtual space of the SoS. Certain digital models are required in order to specify, detect and resolve dependencies among these virtual entities. During the entire SoS lifecycle, these virtual entities are evolving frequently, which makes it even more challenging to handle the architectural dependencies among different SoS systems, subsystems and components. Therefore, reliable approaches and tools are needed for the complexity and change management, as well as prediction of evolution dynamics [3–5].

The Digital Twin (DT) concept provides a method to connect physical and virtual spaces. It was first defined in [6], where a three-dimension DT model was proposed: a DT consists at least three elements, i.e., physical entities in real space, virtual entities in virtual space, and the communications between physical and virtual entities. In recent years, DT has been widely applied in various industrial sectors and the enabling technologies of DT have been evolving rapidly. It reflects the increasing demand of integrating physical systems with their virtual models [7]. A five-dimension DT model was proposed in [8] to promote the DT applications. It is an extended version of the previous three-dimension DT model with two more elements, i.e., DT data and services. As a key enabling technology for Industry 4.0, DT technology has been reshaping the modern manufacturing

✉ Zheng Xiaochen
xiaochen.zheng@epfl.ch

[1] ICT for Sustainable Manufacturing, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne 1015, Switzerland

[2] University of Electronic Science and Technology of China, Chengdu 611731, China

systems from many perspectives such as production planning and control, production process simulation product fault warning, equipment status monitoring and predictive maintenance, layout planning, production index optimization [9, 10].

DTs are expected to support different phases of the system lifecycle, such as design, production, and maintenance [11]. An industrial system may have many DT models across its lifecycle corresponding to different system, subsystems and components. DT models created by different domain experts may have different protocols and standards, which results in heterogeneous structures, in terms of syntax, schema, and semantics. Moreover, DT models evolve frequently across the lifecycle, making them even more difficult to manage. According to a survey about DT applications [7], a universal design and development platform is required to facilitate the integration of different DT models.

When developing an integrated DT platform, semantic modelling is commonly used to capture complex system information in an intuitive way and to provide a concise, high-level description of that information [12]. It formalizes the information using standardized formalism making possible of specifying direct interrelationships among various systems. In addition, a series of tools are available for the design, maintenance, query, and navigation of semantic models, which makes it a promising solution to facilitate the integration of heterogeneous DT models across system lifecycle phases and domains. Previous study [13] has demonstrated the feasibility to build semantics-based DT models using semantic technologies. Researchers [14] make use of semantic models to describe the rules, computing processes and interrelationships related to computing models in order to support automatic decision-makings. Targeting at the manufacturing domain applications, a decision-making DT platform based semantic models is proposed for monitoring and controlling the machining quality [15].

Advanced semantic technologies like Knowledge Graph (KG), have been widely adopted in recent years. KG enables to represent information in a triple format with entities and relationships based on the defined ontology. Moreover, it can be used to derive new knowledge using a reasoner [16, 17]. Previous studies have explored the application of KG in DT development and implementations. It is considered as a main enabling technology for the next generation DT paradigm for linking and retrieving heterogeneous data, as well as descriptive and simulation models [18]. For example, a semantic DT solution was proposed in [19] based on an enterprise KG. It proves that semantic technologies enables to provide a formal representation to reinforce the capability of DT. A knowledge graph model is applied in [20] for integrated knowledge representations for designing data, processing data, inspection data and additional data. Graph-based query languages play a important role for knowledge retrieving with semantic modelling [21]. They support extracting and inferring knowledge from large-scale production data, as well as enable KG queries thus to enhance complexity management of DT models with reasoning capabilities.

When using semantic models, ontology is the basis to support unified knowledge description and DT integration among the specific fields across the entire system lifecycle. Particularly, when using upper-level ontologies such as Basic Formal Ontology (BFO) [22] and Industrial Ontologies Foundry (IOF) domain ontologies, different ontology models can be integrated using a unified format to promote data interoperability. For example, the IOF-MBSE domain ontology is used in [23] to describe co-simulation models based on a standardized artifact representation. Then, using the same ontology, semantic models are used to represent the model structure of verification models in order to realize DT integration [24]. However, the manual construction of ontologies is a time-consuming task [25] which restricts the applications of semantic models. Thus, more efforts on the ontology definitions should be made for managing the complexity of DTs.

From the perspective of the cognitive evolution of IoT technologies, Ahmed [26] proposed the Cognitive Digital Twin (CDT) concept as an augmented digital representation of a physical system, including its subsystems, with certain intelligent capabilities. The authors of [27] categorized DTs into digital twins, hybrid twins and cognitive twins on the basis of their intelligent capabilities. According to this categorization, digital twins are isolated digital models; hybrid twins are interconnected models with integrative prediction capabilities; and cognitive twins are incorporated with cognitive features like sensing complex and unpredicted behaviors, and reasoning for optimization strategies. In a previous study [28], the CDT concept is defined as digital twins that are augmented with semantic capabilities to trace the dynamics of virtual model evolution; to identify interrelationships between virtual models; and to enhance decision-making. This definition emphasizes the critical roles of semantic and knowledge graph technologies for CDT. When developing CDT, ontology plays an important role to support semantic modelling for each digital twin. As demonstrated in previous studies [19–21], semantic modelling and KGs are key enabling technologies to empower cognitive capabilities of DTs thus to realize the CDT paradigm.

CDT represents a promising evolution trend of the current DT technology and is expected to push forward the manufacturing technologies to a higher level of intelligence. Some recent studies and projects have been aiming to apply CDT in different industry sectors and manufacturing domain is one the main focuses. A cognitive twin toolbox conceptual architecture was developed and applied to several use cases such as operational optimization for aluminum production, silicon production, steel and related products production

etc. [27, 29]. An application case was presented in a recent study [30], in which a CDT enabled by KG was developed to support demand forecasting and production planning in a manufacturing plant. A knowledge-driven digital twin was developed in [31] by integrating dynamic knowledge bases with digital twin models to enable intelligent services for autonomous manufacturing such as manufacturing process planning, production scheduling and production process analysis.

Despite the promising future depicted by the CDT concept, many challenges remain to be addressed to realize it. For example, there is a lack of unified reference architectures to support CDT development; there is no available application frameworks to integrate enabling technologies and to provide an implementation tool-chain. This study aims to bridge these gaps by providing a novel solution based on systems engineering and KG to facilitate the integration of heterogeneous models across the entire lifecycle of an industrial system.

The main contributions of this study are: first, to explore the CDT concept using systems engineering methodology; second, to propose a conceptual architecture for CDT design according to existing standards; third, to provide a KG-centric application framework and a tool-chain to facilitate the implementation of CDT; and finally, to verify the proposed framework and tool-chain through a case study.

2 Research methodology

This study follows a systems thinking approach to conduct relevant research activities. Systems thinking is an approach for capturing system nature by analyzing the interrelationships between the components within the system boundary [32, 33]. Based on the systems thinking methodology, the following research steps (RS) are complied:

- **RS 1: Define the scope and scenarios of CDT and provide the definition:** The scenarios are defined based on the experience obtained from relevant research projects and industrial applications (this part is introduced in Sect. 3.1). Correspondingly, the scope (systems boundary) and related concepts about CDT are specified. Stakeholders, as well as their concerns, architecture viewpoints and views extracted from multiple industrial use cases, are adopted to initially identify relevant concepts within the systems boundaries of CDT.
- **RS 2: Identify entities related to the scenarios:** Within the system boundaries, the related entities of each scenario are captured (the part is introduced in Eq. (1)), such as the requirements for constructing CDT [34].
- **RS 3: Specify the interrelationships between entities:** Interrelationships between entities refer to interactions between entities, for example, the traceability between requirement models and verification models (the part is introduced in Eq. (1)).
- **RS 4: Develop an architecture description according to ISO 42010:** The formal architecture description of CDT is developed based on the standard ISO 42010 "Software, systems and enterprise - Architecture processes" (introduced in Sect. 3.2).
- **RS 5: Construct a CDT proof-of-concept:** A prototype of the proposed CDT is constructed based on an industrial use case to illustrate its feasibility (introduced in Sect. 3.3).
- **RS 6: Evaluation of the case study:** Through the case study, the architecture description of the CDT is explained and evaluated (introduced in Sect. 4).

Following the aforementioned approach, the proposed CDT concept, as well as its architecture and application framework are presented in the next section; then, a tool-chain and a case study are provided in the following section.

3 CDT definition, architecture and application framework based on MBSE

3.1 CDT definition

As introduced in previous sections, the CDT concept is evolved from DT, aiming at integrating heterogeneous DT models across the entire lifecycle of a system. It is considered as a subset of DT with additional cognitive capabilities which is proposed for **RS 1** in the research methodology. In a previous study [35], the CDT concept has been initially investigated where it was defined as *a digital representation of a physical system that is augmented with certain cognitive capabilities and support to execute autonomous activities; comprises a set of semantically interlinked digital models related to different lifecycle phases of the physical system including its subsystems and components; and evolves continuously with the physical system across the entire lifecycle*.

It is necessary to clarify that the above definition focuses on the virtual part of the physical-virtual twin to align with existing DT studies. Similar to DT, a complete CDT should also consist of virtual entities and physical entities. To avoid misunderstanding about the names and follow the commons of DT studies, in the rest of the paper, the singular form CDT will be used to represent a complete twin system including both physical and virtual entities. Whereas the plural form CDTs is used to represent multiple different CDT couples corresponding to different application cases.

The main difference between CDT and DT is that cognitive entities of CDT include multiple virtual models across the entire system lifecycle. Each of the models has

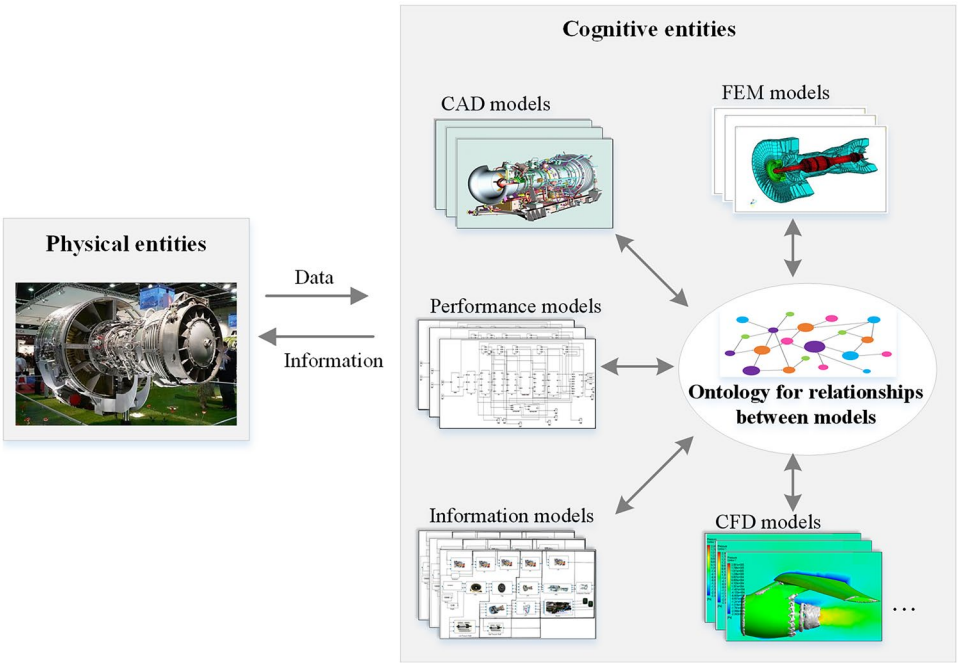

Fig. 1 An example of the CDT concept and its core elements, i.e., physical entities, cognitive entities (including multiple virtual models and Ontology models) and the communications between them

its corresponding ontology descriptions as illustrated in Fig. 1. The ontology of virtual models describes the features of cross-domain models, with the fact that it identifies the interrelationships between different virtual models. As shown in Fig. 1, it supposes the physical entity is an engine; then, the virtual entities may include CAD models, performance models, information models, FEM models, and CFD models etc. These models are used in the different phases of the engine's lifecycle. The ontology is defined as a representational artifact, comprising a taxonomy as proper part, whose representations are intended to designate some combination of universals, defined classes, and certain relations between them [22]. It is developed as the core to formalize the meaning of engine models and interrelationships between all the models.

Based on the previous definitions and systems engineering methodology, for a given system, we formally define its CDT and relevant compositions as follows (**RS 2** and **RS 3** in the research methodology):

$$\begin{aligned} CDT_{sys} = & PE_{sys} \bigcup CE\{\sum Model^{t}(Ms_t, Mp_t, \\ & Mth_t, Ml_t, Mt_t, Mm_t), \\ & Ontology(entities, relationships)\} \\ & \bigcup Comm\{EntitySt, EntityDe, \\ & DType, DContent\} \end{aligned} \tag{1}$$

The meaning of each composition in CDT is explained below:

- CDT_{Sys} refers to the corresponding CDT of a given system *sys*;
- The notation $a \bigcup b$ refers to a collection of a and b. $a = b$ refers to a is equal to b;
- PE_{Sys} is defined as the physical entities of *Sys*;
- $CE\{\sum \text{Model}^{t}(...), Ontology(..)\}$ is defined as one cognitive entity which is a collection of virtual models related to *Sys* with their ontology description;
  - $Model^{t}$ refers to model or data used in the system lifecycle.
  - t refers to a timestamp in system lifecycle which each virtual model is used at.
  - *Ms* (Model Structure) is defined as model topology representing model compositions, interrelationships between them, with entire inputs, outputs and parameters in compositions.
  - *Mp* (Model purpose) refers to the goal for modelling, "why is the model needed?"
  - *Mth* (Modelling theory): the mathematical foundations for modelling, e.g., differential algebraic equations.
  - *Ml* (Modelling language): modelling languages formalizing information and knowledge of the given system which is defined by a consistent set of rules.
  - *Mt* (Modelling tool): tools for developers to build models.
  - *Mm* (Modelling method): a concept to explain the approach to develop models using a given language to represent the system formalisms in one modelling tool, e.g., finite element modelling.

– *Ontology (entities, relationships)* refers to the ontology description representing the model features and topology between them, where one *entity* is defined as one node with the information related to models, such as one model composition. The interrelationships of entities are defined as *relationships* between model compositions which are introduced with details in a previous study [36]:

  - **Reference** — refers to one interaction to track the versions of the related $Model^t$.
  - **Control** — refers to one interaction that $Model^t$ can control another.
  - **Co-simulation** — refers to one interaction that real-time data exchange among $Model^t$ and *sys*.
  - **Model transform** — refers to one interaction that one $Model^t$ is generated from another by model transformation.
  - **Trace** — describes that traceability links of data between different $Model^t$.
  - **Copy** — describes that one $Model^t$ is copied from another.
  - **Refine** — describes that one $Model^t$ is refined by another.
  - **Verify** — describes that one $Model^t$ can verify another.
  - **Satisfy** — describes that one $Model^t$ can satisfy another.

- *Comm{(...)}* is defined as the data and information channels between physical and virtual models or virtual models themselves. Each channel has four key attributes:

  – *EntitySt* (Entities of Start) represents the start of the data and information flow.
  – *EntityDe* (Entities of Destination) represents the end of the data and information flow.
  – *DType* (Data Type) represents the type of data, including real-time data and historical data.
  – *DContent* (Data Content) represents the content been transferred in this data flow.

All of these compositions enable to provide services to stakeholders, data platform and IoT systems.

It is worth mentioning that the term "CDT" in this paper represents the concept of cognitive digital twin in general, whereas in Eq. (1), the "*sys*" subscript is added to indicate that "CDT_{sys}" is a specific cognitive digital twin of the given system "*sys*". This "CDT_{sys}" can be then decomposed to a set of physical entities, cognitive entities and communication entities etc., as shown in Eq. (1).

3.2 CDT architecture

Due to the heterogeneity of the virtual models for different systems, a unified architecture is needed to facilitate the development of CDT. Based on the standard ISO/IEC/IEEE 42010 "Software, systems and enterprise - Architecture processes", a conceptual architecture of CDT is designed (**RS 4** in the research methodology), as presented in Fig. 2. According to this standard, the physical entity *sys* of CDT_{sys} is defined as a "system" with a *physical architecture* which is expressed by an *architecture description*. Using systems thinking, systems can be considered as a material entity or a physical process in the real world. The *architecture description* then identifies *twins-of-interest, stakeholders* and *stakeholders' concerns* respectively. *Twins-of-interest* refers to a collection of cognitive digital twins including cognitive entities and physical entities. *Stakeholders* refers to the individuals, teams and organizations related to the *twins-of-interest*. *Concerns* refers to the system interests related to the *stakeholders*.

The *architecture description* includes *architecture viewpoint*, *architecture view*, *correspondence*, *correspondence rule* and *rational*. The *architecture viewpoint* refers to entities for establishing specifications of constructing, interpreting and using architecture views to frame specific system concerns. The *architecture view* refers to entities for expressing the physical entities from specific concerns. The *correspondence* refers to interrelationships of architecture description entities, such as refinement. The *correspondence rule* refers to specifications for enforcing relations and governing correspondences. The *rational* contains the description, justification or proof of reasoning about the architecture decisions. It supports physical entity development, including the basis for a decision, alternatives, trade-offs and potential consequences of the decisions and reference to sources of additional information. The *architecture viewpoint* includes different model types which are used to develop *virtual models* based on relevant domain specifications.

Within the defined architecture, the *Ontology* in CDT is used to represent *correspondence rule* and *correspondence* among virtual models $\sum Model^t$ as *Comm{(...)}* in order to construct cognitive entities. Moreover, *Ontology* enables to represent *Comm{(...)}* among physical entity *sys* and its models $\sum Model^t$ as well. Finally, the *Ontology* is the basis to support trade-off and reasoning whose outcomes are recorded as *rational*. Through this given framework, the interrelationships between *models* and *Ontology* can be identified clear which is also the reason why the corresponding cognitive entities are required for the physical entities.

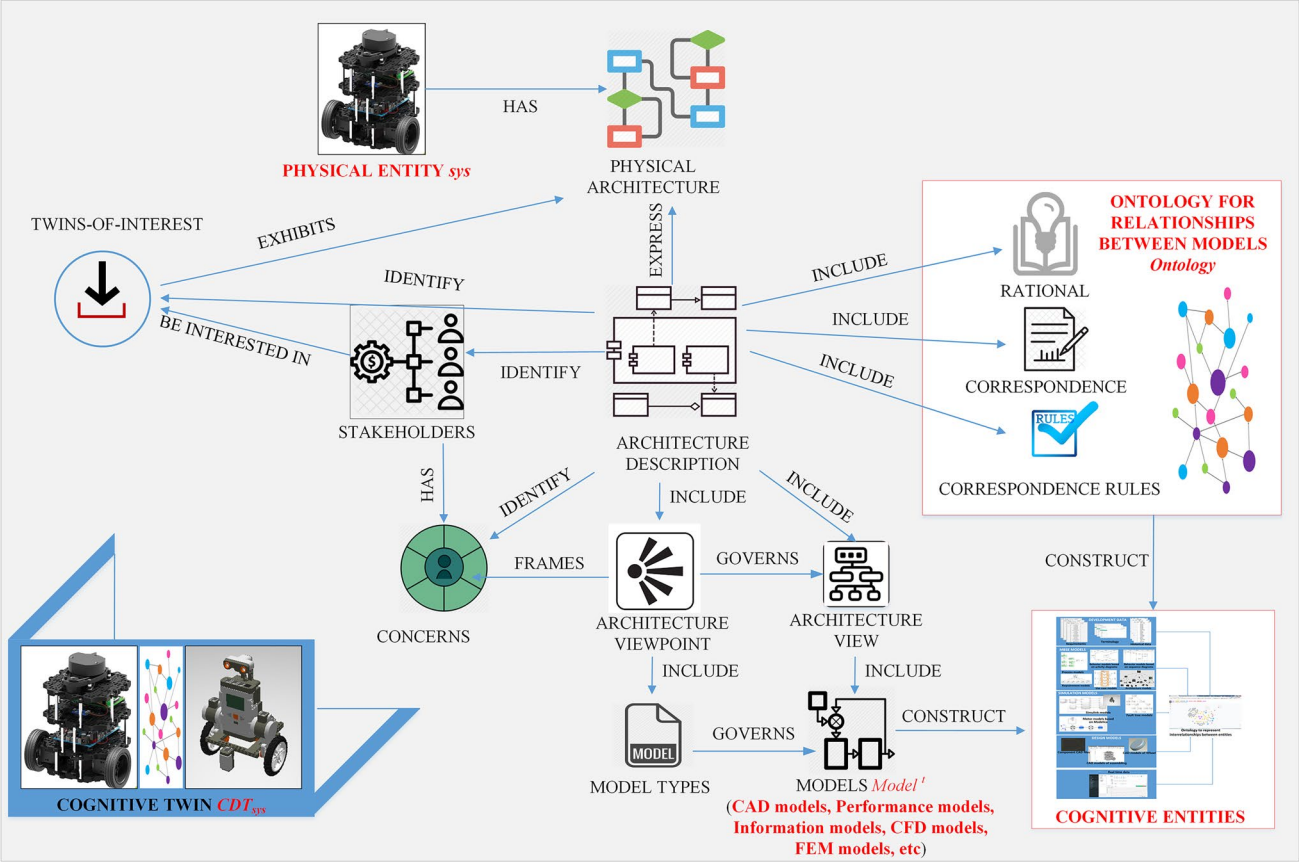

Fig. 2 Conceptual architecture of cognitive digital twins defined based on ISO/IEC/IEEE 42010 standard (labels in red color represent the components defined in Eq. (1))

3.3 CDT application framework based on knowledge graph

The application of CDT in industry is a challenging task as it involves multiple domains and across different lifecycle phases. A framework is developed based on KG to facilitate CDT applications (**RS 5** in the research methodology), as shown in Fig. 3. It is composed of the following five main components:

1. **Industrial system dynamics modelling and simulation**. The purpose of this component is to develop the virtual models for a real physical system *sys* and to simulate the system behaviors based on different models $\sum Model^t$ using modelling and simulation approaches. Most modern industrial systems *sys* are hybridized by continuous and discrete systems [37]. Thus, the virtual models $\sum Model^t$ are required to simulate the hybrid systems, continuous systems and discrete systems providing simulation results for analyzing the system dynamics using a simulation tool *Mt*, which are developed based on mathematical theories related to the physical systems. Finally, simulation results $Model^t$ are generated for constructing CDT_{Sys}.
2. **KG modelling**. KG models are considered as the core to formalize the *Ontology* (topological interrelationships), for virtual models $\sum Model^t$ and the *comm* (communications between physical entities and cognitive entities). Moreover, the KG models are expected to represent the services of each CDT_{Sys}, which refers to the purposes of using the CDT. Based on the basic CDT concepts and domain-specific features of Internet of things [38], we identify seven main concerns when using CDT for industrial systems *Sys*:
   - Social impacts of industrial systems
   - Business models and ecosystems
   - Domain dependent and independent services and application
   - Software architectures of the operational systems and middle-ware, etc.
   - Enabling technologies and systems architecture
   - Security and privacy mechanisms
   - Management strategies of industrial systems

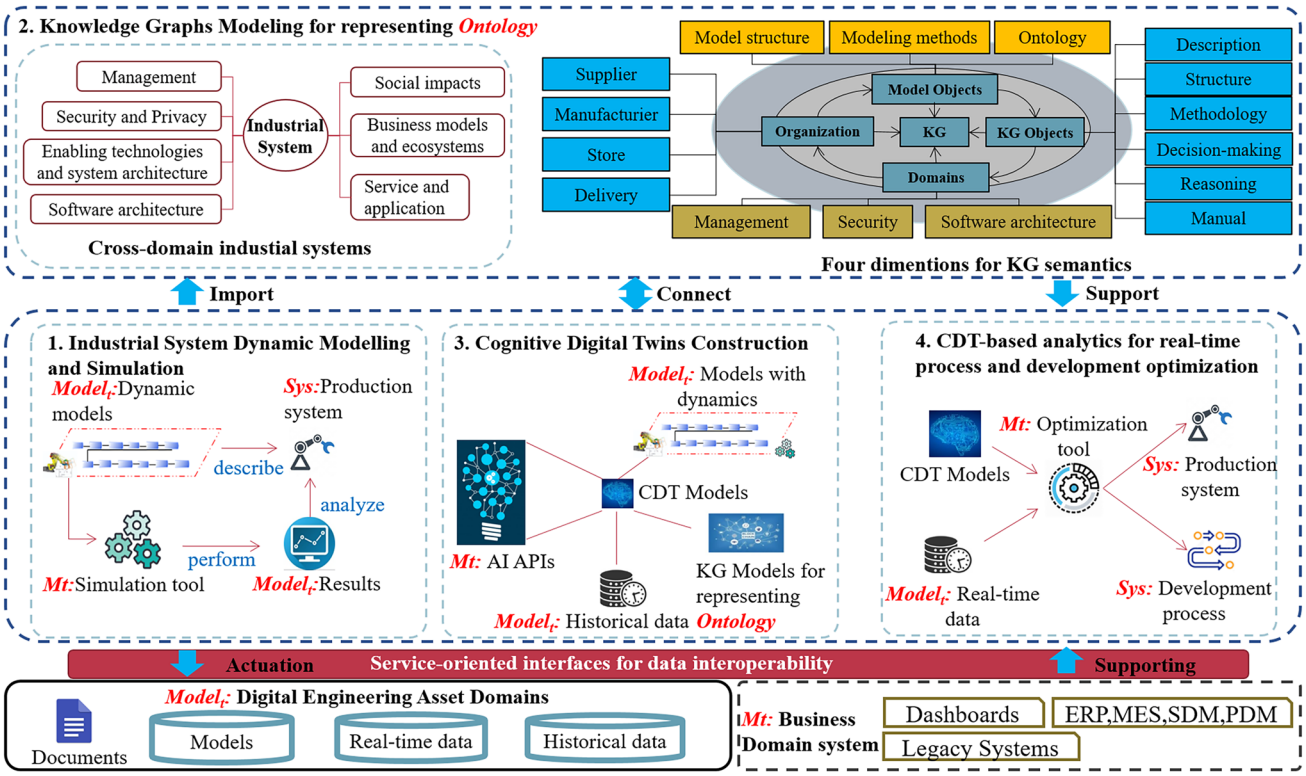

Fig. 3 Application framework of cognitive digital twins enabled by knowledge graphs (labels in red color represent the components defined in Eq. (1))

Based on the seven main concerns, the KG models can be developed in four main dimensions [28]:

- Domains consist of the contents related to system domains including physical entities *Sys* and communications *comm*.
- Model objects refer to the contents related to CDT models CDT_{Sys}, such as virtual models $\sum Model^t$ and *Ontology* (topology between models).
- Organizations represent the organizations related to the system, such as suppliers and manufacturers as stakeholders in Fig. 2.
- KG objects contain the key information for supporting description, structure, methodology, decision-making, reasoning and manuals of knowledge graph models as related architectural viewpoints in Fig. 2.

3. **CDT construction**. *Mt* with machine learning and AI APIs, $\sum Model^t$ including KG models, historical data and results of system dynamics are combined to generate virtual entities of CDT_{Sys}. When developing CDT, machine learning models are trained based on the inputs: (1) KG models, representing *ontology* including domain-specific knowledge and information of virtual models and their topology; (2) Dynamic simulation results, representing the predicted dynamic system behaviors based on simulation models; (3) historical data, representing the previous behaviors of real systems. Then, the generated machine learning models are used to support trade-off and reasoning during development and implementation of the physical system *Sys* in order to obtain the *rational* for its system behaviors.
4. **CDT-based analysis for real-time process and development optimization**. This component is used to make decisions and optimize the *Sys* including physical system and development processes based on the CDT models and collected real-time data $Model^t$. When developing the system *Sys*, CDT can provide decision-makings for designers to select one more expected solution, such as parameter selections and design space exploration [39]. Moreover, CDT enables to optimize the design solutions in order to find an optimal solution. During system implementation, CDT enables to support decision-makings during anomaly detection and forecasting [40]. Moreover, the optimization can be utilized to manipulate the physical entity and control the workflow of system development with better performances as a *Sys*.
5. **Service-oriented interface for data interoperability**. A service-oriented approach are used to support integration of heterogeneous data based on Open Source Lifecycle Collaboration (OSLC) from our previous work

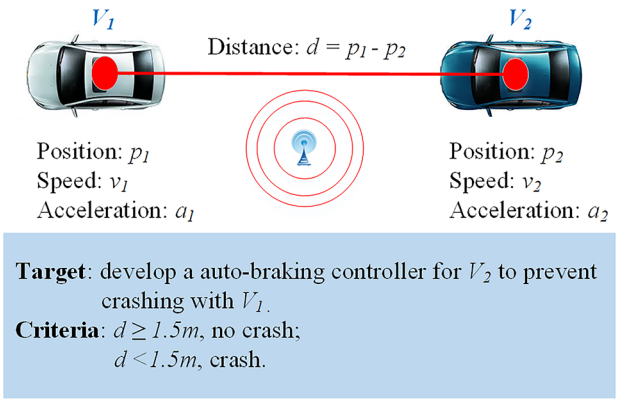

Fig. 4 Scenario definition of the use case for auto-braking controller system

[41]. The digital engineering assets $\sum Model^t$ including models, documents and data across business domains are transformed to unified data formats through the developed OSLC adapters. These unified data is used in different components in our proposed framework to promote the their interoperability. Moreover, different business domain systems such as ERP provide real-time data for making decisions based on CDT.

4 Case study

The aim of this case study is to verify the proposed CDT conceptual architecture and KG-centric framework for CDT application (**RS 6** in the research methodology). This case is designed based on a vehicle auto-braking system development scenario. A tool-chain corresponding to the application framework is provided to enable the CDT development and implementation. A series of simulation experiments are conducted, and a machine learning algorithm is used to create the AI model for constructing CDT for decision-makings in the case study.

4.1 Scenario definition

In this case study, a simplified scenario of the vehicle auto-braking system is defined as shown in Fig. 4. Two vehicles (V_1, V_2) are driving on the same direction with different initial positions (p_1, p_2), velocities (v_1, v_2) and accelerations (a_1, a_2). A controller is expected to be developed for V_2 to protect it from crashing with V_1 in different situations. The distance between them should be more than 1.5 meters, otherwise they are considered as crashed.

The target of the case study is to develop a CDT to support decision-making of the controller development. The CDT is expected to evaluate if the current solution of auto-braking system architecture represented by the system models can satisfy the demands of the auto-braking scenario, i.e., preventing the two vehicles crashing. When developing this controller, a model-based systems engineering approach is adopted in the previous work [42]. KARMA (Kombination of ARchitecture Modeling specificAtion) language is firstly used to support architecture design including requirement modelling, functional modelling, behaviors modelling and physical architecture modelling based on SysML specification [43]. Moreover, based on the code generation approach, KARMA models of physical architecture can be transformed to Matlab language scripts which are used to generate *Simulink* models automatically [44]. Such *Simulink* models aim to simulate the performances of the auto-braking systems and verify the requirements of auto-braking system.

When developing the auto-braking systems using model-based systems engineering (MBSE), KARMA models and *Simulink* models with different parameters represent different solutions. In this case, 100 sets of SysML models and *Simulink* models with different parameters are developed as the solution candidates. The *Simulink* models provide 100 sets of simulation results as the verification of such candidates. In order to make decisions among these solution candidates, a CDT model is expected to analyze the controller solutions (KARMA models) using the previous simulation results. Finally, this CDT model will be used in a web-based process management system to support decision-makings automatically for the controller development [45]. More details about this case are introduced in previous publications as listed in Table 1.

4.2 Tool-chain for CDT development and application

The proposed tool-chain is shown in Fig. 5, corresponding to the application framework. This tool-chain includes *MetaGraph*[1] for system modelling, *Simulink*[2] for verification of the auto-braking system, *Protégé*[3] for ontology construction, and *KNIME*[4] for data processing and AI model training. The workflow of these tools is introduced as follows:

1. A controller for the auto-braking system is designed by system models including requirement models, functional models, logic models and physical structure models in a domain-specific modelling tool *MetaGraph* [43] and verified by virtual simulation models in *Simulink* [42].

[1] A domain-specific modelling tool based on KARMA language, http://www.zkhoneycomb.com/

[2] https://www.mathworks.com/products/matlab.html

[3] https://protege.stanford.edu/

[4] https://www.knime.com/

Table 1 Case study and related previous work

Previous work	Reference	Steps in Fig. 5
A model-driven approach for auto-braking system development	[42]	Steps 1–3
KARMA Language supporting architecture design of auto-braking system	[43]	Step 1
KARMA Language supporting code generation for implement *Simulink* models automatically	[44]	Step 2
GOPPRRE ontology generation from KARMA language for constructing knowledge graph models	[46]	Step 4
A process management platform which can integrate CDT for selecting parameters automatically	[39]	Step 8

Such *Simulink* models can be generated from system models using the code generation function in *MetaGraph*. The detailed models built in this case study are listed in Table 2.

2. Ontology models are generated from KARMA models of auto-braking system architectures which represent the topology and information related to the *MetaGraph* models and *Simulink* models [46]. Thus, one set of system models refers to one entire solution of the auto-braking controller development. Moreover, the generated *Simulink* models are used to verify the solutions and provide their results for controller performances.
3. Several sets of system models and related simulation results obtained from *Simulink* models are imported to *KNIME* for data processing [47]. In *KNIME*, a data-analysis workflow is developed to capture the required data from ontology models, to develop machine learning algorithms for AI model training and generation, and to validate the AI models based on the captured data.
4. Finally, after the AI models are generated, they are integrated with a process management system in order to support parameter selection for system developers.

4.3 Virtual model development for CDT construction

A set of the KARMA models are developed to define requirements, functions, behaviors, physical structure and verification of the auto-braking system based on the SysML specification. Such KARMA models are constructed as one solution for the auto-braking system development. As introduced in the *Scenario definition*, each set of KARMA models representing one solution has their own parameters[5]. Moreover, such models can be used to generate a *Simulink* model for verifying the performance of the designed controller.

4.3.1 Architecture models

As shown in 6, some examples of the KARMA models are created with relevant information listed in Table 2. The *physical entities*, *cognitive entities* and *comm* construct a complete CDT, which enables to make decisions in the process management system [45]. The overall architecture design and verification process for the auto-braking system consists of the following five phases:

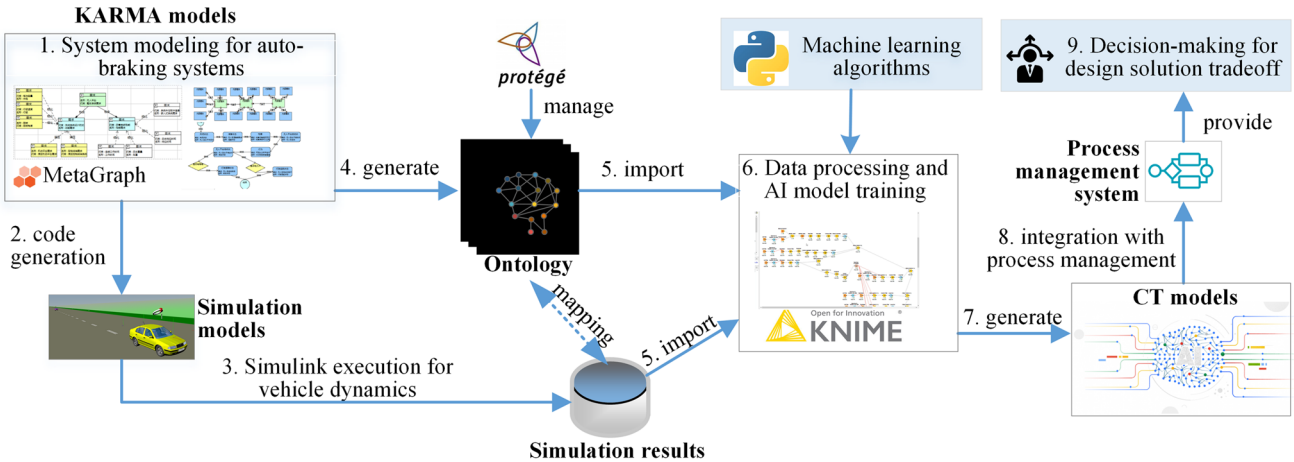

Fig. 5 Tool-chain for CDT development and application

[5] The entire models are introduced in https://www.youtube.com/watch?v=FlccxJBdtwo

Table 2 Cognitive digital twins constructed for the case study

Entities		Models	Views
Physical entities		Decision-making processes for auto-braking system design	Making decisions in the process management system for auto-braking systems
Cognitive entities	Models (Virtual entities)	Requirement models	SysML requirement diagram for formalizing the requirements of auto-braking systems
		Function models	SysML use case and activity diagrams for developing the functions of the auto-braking systems
		Behavior models	SysML State machine diagram and Sequence diagram for behavior formalism of auto-braking systems
		Physical structure models	SysML Block definition diagram and Internal block diagram for describing the system structure of the auto-braking systems
		Verification models (mirror to Simulink models which are not included in models)	SysML Parametric diagram and Internal block diagram to describe the parameter configuration and model structure of Simulink models
		Simulink models	Simulation models for verifying the controller performances
		Simulation results	Simulation results obtained from Simulation models for verifying the controller performance
	Ontology	KG models generated from SysML models described in OWL	Topologies between all the model entities with their own information
Comm		Integration of CDT models and a web-based process management platform	Implementations of Decision-makings between system development process based on cognitive entities

- **Requirement**: KARMA models are used to create SysML Requirement diagrams for describing the requirements of the controller when developing the auto-braking system. As shown in Fig. 7A, requirements such as "top level requirement for auto-braking system", "the system shall provide a base brake funnctionality where the driver

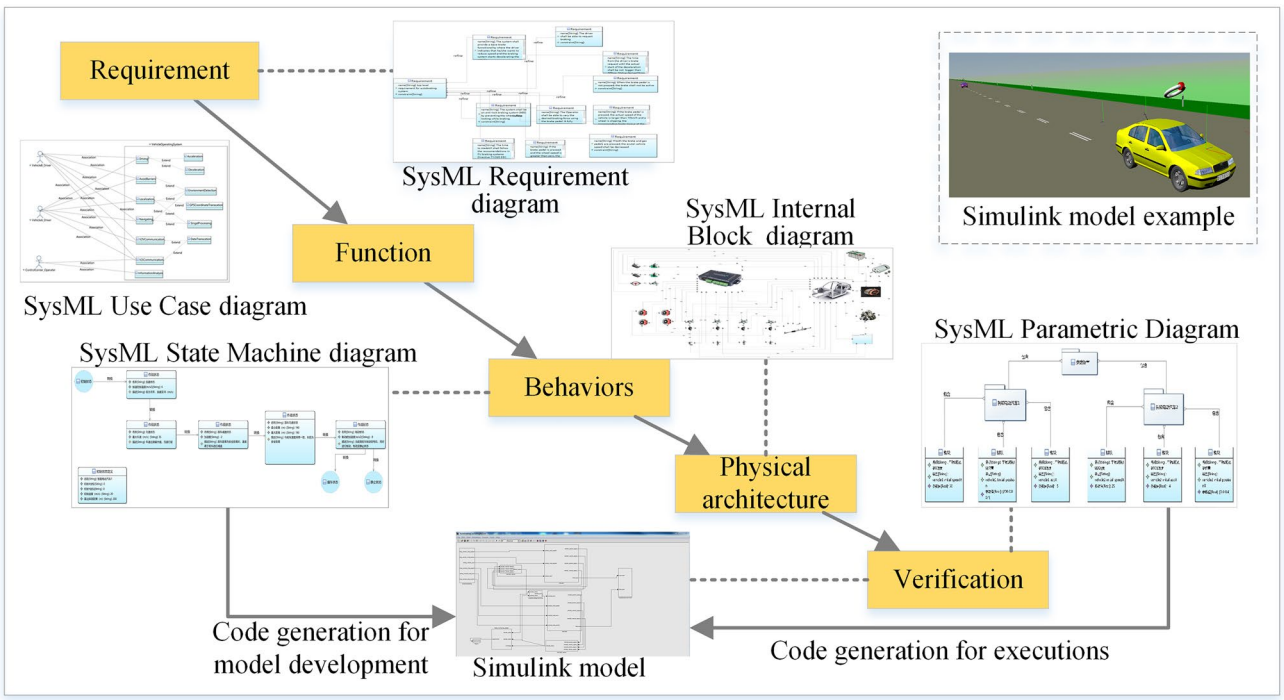

Fig. 6 Construction of the CDT virtual models including architecture models and *Simulink* models

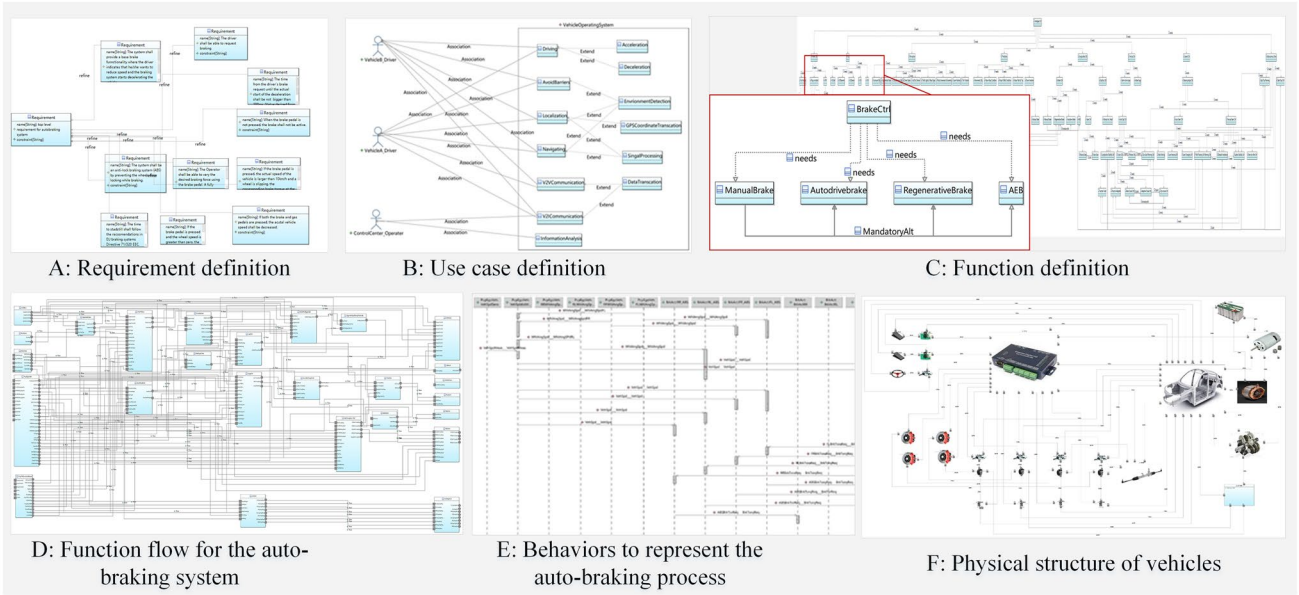

Fig. 7 KARMA models in MetaGraph 2.0

indicates that he/she wants to reduce speed and the braking system starts decelerating the vehicle", are defined in the Requirement Diagram.

- **Function**: SysML Use Case and Activity diagrams are used to develop KARMA models aiming to identify the use case scenarios and function flow of the controller. As shown in Fig. 7B, several Use Case diagrams are created to identify the stakeholders and the features (an abstract concept of a collection of functions). Such features are decomposed into 158 functions as shown in Fig. 7C. For example, the function of breaking control needs four functions. Such functions are then defined as an entire function flow, as shown in Fig. 7D, which is used to represent the functioning process for the entire auto-braking scenario.
- **Behaviors**: SysML State Machine diagram and Sequence diagram are used to develop KARMA models for designing the logic flow of the controller with the behaviors of each component. As shown in Fig. 7E, system behaviors of each component are represented by Sequence diagrams in order to identify all the physical components in the physical structure.
- **Physical structure**: SysML Definition Block diagram and Internal Block diagram are used to develop KARMA models for describing physical structure of the auto-braking system including its components, such as the controller. As shown in Fig. 7F, the entire physical structure is shown including all the related system components.
- **Verification**: In order to realize automated testing from the architecture models, we construct a KARMA model to derive Simulink models for simulation based on SysML Internal Block diagram. Moreover, KARMA models based on SysML Parametric diagram are defined to support parameter settings for automated testing as shown in Fig. 8.

It is worth noting that the decision-making process for the auto-braking system design is considered as the physical entity in this case study. The main reason is that from the systems engineering perspective, a process can also be considered as a system. In this case study, although the targeted "physical" entity is the auto-braking system, we are focusing on the design phase of its lifecycle, or more specifically the decision-making process among different design solutions. Therefore, the system becomes process-oriented instead of product-oriented. The physical auto-braking system becomes the target of the process and more other information related to the decision-making process itself is included to create a more complete "system". Corresponding to the decision-making process, different types of models such as requirement models, function models, behavior models, and physical structure models are defined as the virtual entities as introduced above. Among them, the physical structure models can be mapped to the physical auto-braking system which can be used in the following lifecycle stages such as production and maintenance.

4.3.2 Code generation for automatic testing

Enabled by the code generation function of KARMA [45], the KARMA models of SysML Internal Block diagram are used to generate the *Simulink* models. First, a KARMA

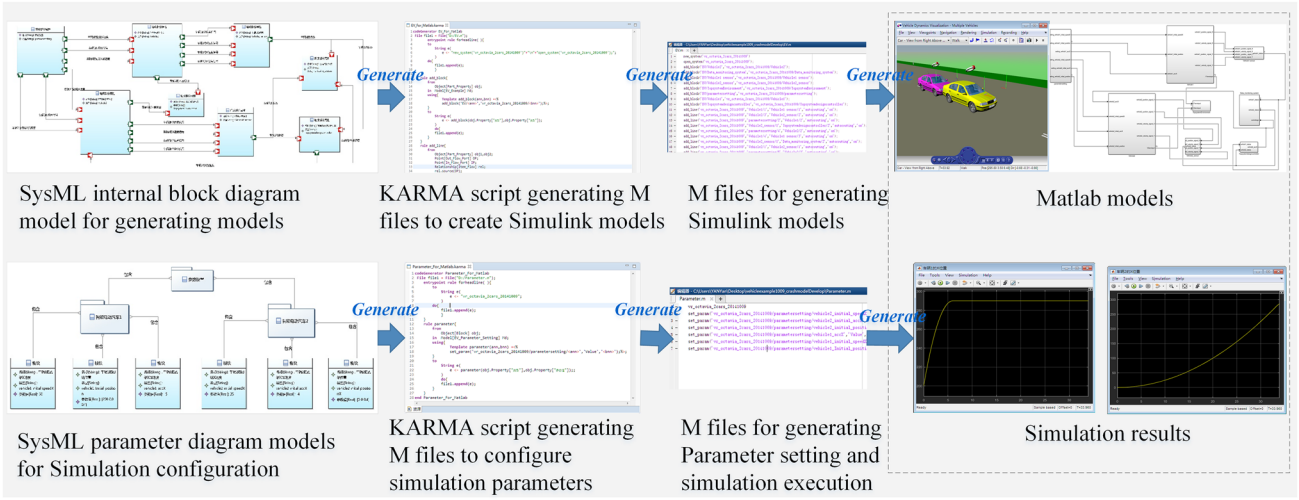

Fig. 8 Code generation process from KARMA models for Simulink execution

script is designed in order to implement code generation. As shown in Algorithm 1, after identifying all the elements and connections of the KARMA model which represents the physical architecture, an M file is generated based on the developed KARMA script. Then, an M file, which can be used by Matlab, is generated from KARMA Internal Block diagram model for generating Simulink Models finally.

Algorithm 1 Generating M Files for generating the Simulink model Using Code Generation

Input: A KARMA model a of SysML internal block diagram for defining physical architecture

Output: An M file α for generating a Simulink model

Initialisation: import a
for *elements* in a **do**
    % identify all the model elements
    if (an *element* is captured **then**
        print ("add_block", element's property);
    end if
    % print a function for adding a element in the M file α
end for
for *connections* in a **do**
    % identify all the connections in a
    if (a *connection* is captured **then**
        print ("add_line", *connection*'s property (*connection*'s name in Simulink), endPoint'property (end point of the *connection*), startPoint'property (start point of the *connection*));
    end if
    % print a function for adding a connection in the M file α
end for
    return α

Algorithm 2 Generating M Files for simulation configuration Using Code Generation

Input: A KARMA model b of SysML parametric diagram for defining parameter configuration during verification

Output: An M file β for configuring the Simulink model and running the simulation

Initialisation: import b
for *constraints* in b **do**
    % identify all the *constraints*
    if (a *constraint* is captured **then**
        for *parameters* connected to the *constraint* **do**
            if (a *parameter* is captured **then**
                print ("set_param" ,parameter's property);
            end if
            % print a function for setting a parameter of the block of Simulink in the M file β
        end for
    end if
    print ("sim" ,constraint's property);
    % print a function for runing the simulation in the M file β
end for
    return β

Moreover, KARMA models of SysML Parametric diagrams are used to describe parameter configurations of *Simulink* model executions. As shown in Algorithm 2, after each constraint is identified, based on the connected parameters, the M script is printed for configuring the parameters in Simulink based on the KARMA script. Then, it enables to generate an M file to execute parameter setting and simulations through code generation. Through these KARMA models, Matlab scripts are generated to configure the *Simulink* models and execute the simulation automatically. During the automatic testing process, 6 key parameters are captured when implementing the simulations:

- initial position of the V_1
- initial velocity of the V_1
- initial acceleration of the V_1
- initial position of the V_2
- initial velocity of the V_2
- initial acceleration of the V_2

The KARMA model based on SysML Parametric diagram describes the given range of each previous parameter. Then, when executing the simulations, the value of each parameter in Simulink is set for every simulation.

4.4 Machine learning algorithm for CDT construction

In order to construct CDT, 100 KARMA models based on SysML specifications are firstly transformed to 100 OWL files by *MetaGraph*. Moreover, *Simulink* models generated from SysML models are implemented with 100 simulation results. As shown in Fig. 9A, the OWL models and simulation results from *Simulink* models are input into *KNIME* for developing AI models used in the web-based process management system. As shown in Fig. 9C, the AI model aims to import the OWL file (model structural data generated from KARMA models) to provide a decision-making option (being crash or not) for the process management system. In order to train the AI model, the OWL files are transformed to structural OWL data using SPARQL query [48] and simulation results are labelled as (0 (crash) and 1 (no crash)) based on the situation if the distance of these two vehicles is less than 1.5 meters anytime. These labels are mapping to the structural OWL data which means if the OWL data representing each solution from KARMA models can satisfy the demand that these two vehicles cannot be crashed.

The data from labels and structural OWL data are collected as training dataset for a five-layer neural network model development empowered by the APIs provided by the Tensorflow in *KNIME*. Some key parameters of the neural network model are listed in Table 3. Regarding the training dataset, 80% of the simulation data (80 pairs of OWL structural data and labels (1 or 0)) are used to train the neural network model and the rest 20% are used to test the performance of the obtained model.

Algorithm 3 Data Processing and AI model training

Input: Ontology OWL files, *Simulink* simulation results

Output: Prediction model based on SysML models

Initialisation: import OWL_data, $Simulink\_data$,
for $data_i$ in $Simulink\_data$ **do**
  find $result_i = (v1_i, v2_i, p1_i, p2_i, t_i)$
  if ($result_i$ meets $crash$ criteria) **then**
    $label_i = 0$
  end if
  if ($result_i$ doesn't meet $crash$ criteria) **then**
    $label_i = 1$
  end if
  $Label\_data = [Label\_data, label_i]$
end for
$X, Y = OWL\_data, Label\_data$
Define $batch = M, epochs = N, learning\_rate = \alpha, layers = L$
$(W, b) =$ initialized weights
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    $A^0 = x$
    for $l = 1$ to L **do**
      $Z^l = W^l * A^{l-1} + b^l$
      $A^l = sigmod(Z^l)$
    end for
    $y^\wedge = A^l$
    $J = -(y * ln(y^\wedge) + (1-y) * ln(1-y^\wedge))$
    $\partial J/\partial A^L = -y/A^L + (1-y)/(1-A^L)$
    for $K = L$ to 1 **do**
      $\partial J/\partial Z^k = (\partial J/\partial A^k) \odot (dA^k/dZ^k)$
      $\partial J/\partial A^{k-1} = W^{k^T} * (\partial J/\partial Z^k)$
      $\partial J/\partial W^k = (\partial J/\partial Z^k) * A^{k-1^T}$
      $\partial J/\partial b^k = sum(\partial J/\partial Z^k, axis = 1)$
      $W^k = W^k - \alpha * (\partial J/\partial W^k)$
      $b^k = b^k - \alpha * (\partial J/\partial b^k)$
    end for
  end for
end for
return $Y^\wedge =$ predict label probability

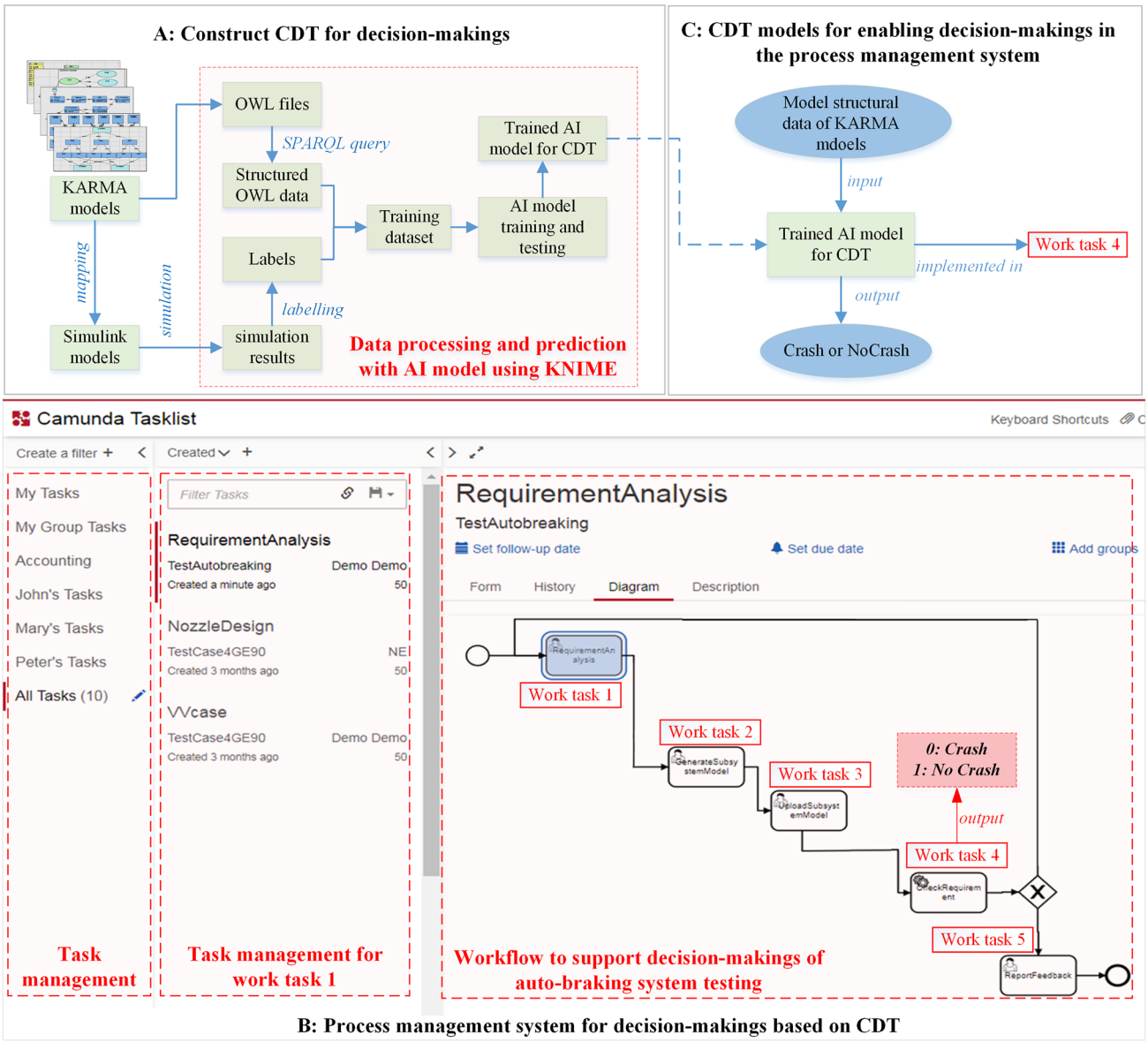

Fig. 9 CDT models supporting automated parameter selections

The details of the data processing are demonstrated in Algorithm 3. The source code of the data processing and neural network model training are available online[6] with all the data samples. One set of the data contains one OWL file and multiple simulation result files generated with *Simulink*. Some key parameters and performance indicators of the neural network model are listed in Table 3. With 80 sets training of data, the five-layer neural network produced 65% accuracy with an 86% recall rate among 20 sets of testing data. The accuracy of the neural network is relatively low mainly because the size of the data samples is small. Since the machine learning algorithm is not the main contribution of this paper, we used a basic neural network structure to demonstrate the workflow. In real industrial applications, much larger training data size will be available and more advanced machine learning algorithms can be applied to achieve a better performance.

4.5 Applying CDT for supporting decision-making

As shown in Fig. 5, the trained neural network model based on historical data referring to the CDT is integrated into a web-based process management system for the auto-braking

[6] https://github.com/zhengxiaochen/cognitivetwins

Table 3 Parameters and performance of the neural network model

Neural network model		Performance	
Layers	5	Accuracy	0.65
Batch	30	Recall	0.86
Learning rate	0.001	TN(FN)	7 (1)
Epochs	1000	TP(FP)	6 (6)

system development (Fig. 9B). This development process is developed in our previous research [39] aiming at selecting parameters for co-simulation of auto-braking system design. In the previous research, a web-based process management system is developed to select parameters for co-simulation based on the Simulink simulation. In this paper, we develop a CDT which is plugged into the web-based process management system to provide a decision-making process for automatic testing. The whole process includes 5 tasks:

- Work task 1, referring to requirement analysis of the auto-braking system. In this working task, requirement models are developed using KARMA. Through the web-based process management system, users can reach to the requirement models developed in MetaGraph 2.0.
- Work task 2, referring to generating architecture model for auto-braking systems. In this working task, function, behavior, physical structure and verification models are reviewed. After the models are developed, different stakeholders enable to reach to the related KARMA models through the web service.
- Work task 3, referring to uploading models for auto-braking system. In this working task, KARMA models are generated into OWL models and uploaded to the server which is used for decision-makings through CDT.
- Work task 4, as shown in Fig. 9C, when the CDT is integrated with the work task 4, the web-based process management system implements the CDT through a Java program which captures the inputs from the OWL models and then provides the decision-making options if the two cars are crashed or not without simulations.
- Work task 5, referring to feedback. In this working task, feedbacks based on the decision-makings from the CDT are provided to the users in the web-based process management system.

4.6 Summary of case study

In the above case study, in order to design an architecture description for the auto-braking system design process, the architecture design and verification of the auto-braking system are considered as two separate architecture viewpoints. Therefore, the KARMA models and Simulink models are defined as model type to support the architecture description of such process. They are essential elements for constructing the virtual entities. It is worth to mention that the KARMA models for verification are not considered as virtual entities, because they are only used to generate Simulink models. The simulation results generated from Simulink models are considered as virtual entities in the scope since they are important to the decision-making process.

The M files for Matlab and KARMA scripts for code generation are not considered as virtual entities. The reasons are two-fold: first, they are introduced to generate Simulink models from KARMA models and have no specific descriptions about the real auto-braking system design process; thus, they are not included in any architecture view; second, they are not defined in the ontology models; thus, there is no relationships defined between them and other virtual entities such as architecture models, simulation models and simulation results.

The OWL models generated from KARMA models contain specifications of the topology between virtual entities. They include the information of KARMA models for verification which is the mirror to Simulink models, thus, it is not necessary to add extra information to represent Simulink. The simulation results are not included in the generated OWL models. This information is added to the generated ontology models manually using Protégé. In summary, the OWL models include the following three types of information:

- KARMA models including requirement, function, behavior and physical structure models and their topology.
- KARMA models including verification models which are mirrored to Simulink models and their topology.
- Simulink results which are linked to Simulink models and their topology.

All the cognitive entities and their quantities in the scope of the case study are listed in Table 4. There are totally 31 KARMA models, including requirement, function, behavior, physical structure and verification models. Among them, the SysML parametric diagram model is configured 100 times with different settings to generate the Simulink models and the other 30 remain the same. Therefore, 130 KARMA models are created for this case study. Simulink models and results are separately generated by 100 times; thus, their numbers are both 100. Ontology models are generated by 100 times in order to synchronize with Simulink models. A dedicated plugin is developed to integrate these virtual entities into the web-based process management system.

When constructing the cognitive entities, model evolution, understanding the interrelationships, and enhancing the decision-makings are three critical features. They are further elaborated as follows:

Table 4 CDT entities and their quantities

Physical entity		Decision-making process during the auto-braking system design	Quantities
Cognitive entity	Models (Virtual entities)	KARMA models supporting auto-braking system design, including requirement, function, behavior, physical structure model	130
		Simulink models supporting auto-braking system verification which are mapped to KARMA models for verification	100
		Simulation results from Simulink models	100
	Ontology	Ontology models developed in Protégé representing topology among KARMA architecture models, Simulink models and simulation results	100
Comm		A plugin for the web-based process management system embedded the CDT to implement decision-makings without simulation execution [45]	1

First, model evolution is defined from two aspects: (1) dynamic model evolution across the entire lifecycle; (2) model evolution based on each model baseline. The first aspect refers to dynamic model evolution from problem domain to solution domain across the entire system lifecycle. In the case study, we develop requirement models, function models, behavior models, physical structure models, verification models and simulation models using KARMA. These models support the whole development process from problem domain (black box) to solution domain (white box). All these KARMA models are transformed to ontology models including all the evolution information from requirement to verification. The second aspect is model evolution based on its baseline. Because all the model topology and parameters are described using ontology models, the IT techniques enable to support version management, change management and consistency management of KARMA models. Using such techniques, each KARMA model can be managed from its baseline and the following versions.

Second, using a unified ontology, virtual entities, physical entities and their relationships can be formally described. KARMA language [43] and GOPPRRE ontology [46] provide a basic specification to develop architecture models. Meta-models, such as SysML diagrams, are developed under a unified semantic data structure which promotes the interoperability of architecture models. Through code generations, KARMA architecture models can be transformed to Simulink models which describe the transformation rules from architecture to simulation models. With top-level ontology, such as BFO, ontology models enable to describe the interrelationships between physical systems, system lifecycle, architecture models, models in other simulation tools and other data formats.

Third, the ultimate objective of the developed CDT is to replace the simulation execution in order to improve the efficiency of the decision-making process. When using traditional decision-making processes, Simulink models are created through code generation and then executed to obtain the given simulation results as the input of the decision-making algorithm, which then provides a final decision reference. After constructing the CDT with pre-trained AI models, architecture model information in OWL models is provided as input of the decision-making algorithm directly. The code generation and simulation execution are not required compared with the traditional process. In this situation, the efficiency of decision-making process can be promoted.

5 Discussion

5.1 Main achievements

As an emerging concept, CDT is still in its early stage of development. Although some previous studies have explored the CDT concept from theoretical perspective, there is still a lack of successful application cases to realize the concept. This paper aims to bridge the gap between CDT theoretical concept and industrial applications by providing a conceptual architecture and an application framework containing necessary enabling technologies and tools.

A case study about auto-braking system development is used to validate the proposed solution. The CDT of this system is developed based on ontology models, heterogeneous virtual models and system dynamics simulation results. In this case, the development process of the auto-braking system can be considered as the physical entity of the CDT. As shown in Table 2, all the defined concepts related to the decision-making process for the parameter selection are captured. Using the KARMA language and *Simulink* tool, seven types of models are used as virtual entities. Moreover, the OWL models generated from KARMA models are used to represent model information and the topology across models. All these models compose the cognitive entities of the CDT. The communication between physical entities and cognitive entities refers to the integration of the AI models and the web-based process management system. The results of the case study prove that the given tool-chain is capable of constructing a practical CDT.

The conceptual architecture of the CDT is also validated through the given case study. In this case, the architecture description of the physical entity, i.e., the development process of the auto-braking system, includes viewpoints conforming concerns from stakeholders: (1) requirement analysis; (2) function definition; (3) logic flow of each components in the auto-braking system; (4) physical structure; (5) verification of the controller; (6) performance analysis of the controller. *Simulink* models and KARMA models are used to represent views governed by such viewpoints. Because of code generation from KARMA models, the information related to *Simulink* models is represented in KARMA models. Thus, the OWL models which are generated from KARMA models, enable to represent the information of not only KARMA models themselves, but also *Simulink* models and topologies among them. Such OWL models can be considered as ontology to represent relations and correspondences. The OWL models are constructed based on its own schema which can be defined as correspondence rules.

In the case study, the feasibility of the proposed toolchain is also validated corresponding to the proposed CDT application framework. First, *Simulink* is used to implement the system dynamic analysis and simulation for constructing CDT. Second, OWL models generated from the KARMA models are constructed for KG modelling. Though the given KARMA models in the case study only represent seven viewpoints related to part of the mentioned concerns, the KARMA language enables to be extended with other meta-models for further concerns. *KNIME* is used to integrate AI platforms, OWL models and data from *Simulink* model execution in order to construct CDT models. Finally, the CDT models are integrated with a real web-based process management system in order to support decision-makings of parameter selections during auto-braking system development.

5.2 Limitations

The work presented in this paper can be considered as a primary demonstrator to reveal the great potential of the CDT concept. However, many advanced enabling technologies are required to fully realize the CDT vision, such as semantics engineering and KG, machine learning, IoT, and cloud computing. Due to limited resources, the case study mainly focused on some of the main functions of the CDT concept. There exist several limitations of this study as listed below.

- Due to limited resources, a simplified use case is used with much less factors than real complex systems. This impacts the performance of the obtained decision-support model. Moreover, since the machine learning algorithm is not the main focus of this paper, limited efforts are spent on the parameter tuning of the algorithm. In practical applications, much more data will be collected and more advanced machine learning algorithms should be designed to obtain better performance.
- The case study only covers the development phase of the auto-braking system, whereas a complete CDT is expected to include more lifecycle phases, such as production, maintenance, and recycling. To construct such a complete CDT, a special team involving experts from all relevant domains is needed, which requires high-level of inter-organizational interoperability.
- The automatic testing scenario based on Simulink is limited by a simplified model. In this paper, the purpose of the use case is to evaluate our CDT concept rather than a real verification for the auto-braking system design. Thus, more complex models such as CAE models, CAD models are not considered in the case study. Further research will be implemented for the industrial case to improve the model complexity.
- The service-oriented interfaces are not included in the case study. However, previous research [41] has provided a possible solution for integrating heterogeneous data using OSLC, which will be integrated in the future research.

5.3 Future works

CDT is a novel concept which is believed to be the next evolution of DT. The contributions of this paper, including the CDT concept, conceptual architecture and application framework, paved way to more future research opportunities. Some of them are listed below:

- Application of advanced KG and relevant technologies in CDT development: Semantic modelling and KG technologies are key enabling technologies for CDT. They are currently under rapid development and new tools are appearing frequently. It is important to investigate such new achievements and explore their applications for CDT development.
- Development and application of standardized industrial ontologies: Ontology is crucial to obtain high interoperability among digital entities. However, most ontologies are developed based on their own scenarios without standardization. Top-level ontologies, such as BFO and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [49], can help increase the semantic interoperability among different lower-level ontologies. Efforts are needed to investigate how to unify and standardize existing domain ontologies based on such top-level ontologies.
- Development and application of MBSE technologies in complex system development: MBSE models provide a

structural description for formalizing DTs. Such models are the foundation to formalize the topology between different digital entities. The semantic and unified languages supporting MBSE, such as KARMA language, provide a potential way to create KGs automatically, which can accelerate the CDT development.

- Application of advanced machine learning techniques to support CDT implementation: The performance of the adopted machine learning algorithms determines the reliability of the decision-making results of CDT. More efforts are required to explore how to select and apply the optimal algorithms for CDT implementations.
- More detailed industrial CDT implementation reference architectures and tool-chains: This paper presents a conceptual architecture and tool-chain based on the knowledge of the authors. Experts from other domains might have different viewpoints and concerns about CDT implementation. More architectures and tools are required to complete the CDT vision.

6 Conclusion

This paper proposes a formal definition of CDT based on a systems engineering approach. Moreover, a CDT conceptual architecture is defined based on the systems engineering standard ISO 42010. To facilitate CDT development and implementation, a KG-based framework is developed together with an enabling tool-chain. To verify the proposed framework and tool-chain, a case study of an auto-braking system development is conducted. KARMA models and *Simulink* models are used to define solutions and verify requirements. Based on such models, a multi-layer neural network is trained based on simulation data and ontology models generated from KARMA models, which is then utilized to support decision-making. The case study demonstrated the practicability of the proposed CDT concept, architecture and reference framework. The simulation results indicate the potential of CDT in promoting the decision-makings during complex system development. This study bridges the gaps between theoretical CDT concept and industrial CDT applications. It reveals the great potential of CDT, as the next generation of DT, for complex system development and management.

Author Contributions L.J. and Z.X. conceived the presented idea and developed the theory and performed the computations. Y.Z. and W.J. provided support for the case study. D.K. encouraged L.J. and Z.X. to investigate the topic and supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

Funding Open access funding provided by EPFL Lausanne The work presented in this paper has been partially funded by the EU H2020 project FACTLOG (869951) - Energy-aware Factory Analytics for Process Industries, and EU H2020 project QU4LITY (825030) - Digital Reality in Zero Defect Manufacturing.

Availability of data and material The data that support the findings of this study are available from the corresponding author upon request.

Code availability The codes that support the findings of this study are available from the corresponding author upon request.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

Competing interests The authors declare no competing interests.

References

1. Tao F, Qi Q (2017) New it driven service-oriented smart manufacturing: framework and characteristics. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49(1), 81–91, DOI: 10.1109/TSMC.2017.2723764
2. Kumar P, Merzouki R, Bouamama BO (2017) Multilevel modeling of system of systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48(8):1309–1320. https://doi.org/10.1109/TSMC.2017.2668065
3. Fortino G, Russo W, Savaglio C, Shen W, Zhou M (2017) Agent-oriented cooperative smart objects: From iot system design to implementation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48(11):1939–1956. https://doi.org/10.1109/TSMC.2017.2780618
4. Mordecai Y, Orhof O, Dori D (2016) Model-based interoperability engineering in systems-of-systems and civil aviation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48(4):637–648. https://doi.org/10.1109/TSMC.2016.2602543
5. Tavčar J, Horvath I (2018) A review of the principles of designing smart cyber-physical systems for run-time adaptation: Learned lessons and open issues. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49(1):145–158. https://doi.org/10.1109/TSMC.2018.2814539
6. Grieves M (2014) Digital Twin: Manufacturing Excellence Through Virtual Factory Replication. Tech Rep. https://doi.org/10.5281/zenodo.1493930
7. Qi Q, Tao F, Hu T, Anwer N, Liu A, Wei Y, Wang L, Nee A (2019) Enabling technologies and tools for digital twin. J Manuf

Syst. https://doi.org/10.1016/j.jmsy.2019.10.001, https://linkinghub.elsevier.com/retrieve/pii/S027861251930086X
8. Tao F, Zhang M, Cheng J, Qi Q (2017) Digital twin workshop: a new paradigm for future workshop. Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS. https://doi.org/10.13196/j.cims.2017.01.001
9. He B, Bai KJ (2021) Digital twin-based sustainable intelligent manufacturing: A review. Adv Manuf 9(1):1–21
10. Kritzinger W, Karner M, Traar G, Henjes J, Sihn W (2018) Digital twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine 51(11):1016–1022
11. Tao F, Zhang H, Liu A, Nee AYC (2019) Digital Twin in Industry: State-of-the-Art. IEEE Trans Ind Inf 15(4):2405–2415. https://doi.org/10.1109/TII.2018.2873186, https://ieeexplore.ieee.org/document/8477101/
12. Hui W, Dong X, Guanghong D, Linxuan Z (2007) Assembly planning based on semantic modeling approach. Comput Ind 58(3):227–239. https://doi.org/10.1016/j.compind.2006.05.002
13. Kharlamov E, Martin-Recuerda F, Perry B, Cameron D, Fjellheim R, Waaler A (2018) Towards semantically enhanced digital twins. In: 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp 4189–4193. https://doi.org/10.1109/BigData.2018.8622503, https://ieeexplore.ieee.org/document/8622503/
14. Zheng P, Sivabalan AS (2020) A generic tri-model-based approach for product-level digital twin development in a smart manufacturing environment. Robot Comput Integr Manuf 64:101958. https://doi.org/10.1016/j.rcim.2020.101958, https://www.sciencedirect.com/science/article/pii/S0736584519305289
15. Liu S, Lu Y, Li J, Song D, Sun X, Bao J (2021) Multi-scale evolution mechanism and knowledge construction of a digital twin mimic model. Robot Comput Integr Manuf 71:102123. https://doi.org/10.1016/j.rcim.2021.102123, https://www.sciencedirect.com/science/article/pii/S0736584521000090
16. Ehrlinger L, Wöß W (2016) Towards a definition of knowledge graphs. In: CEUR Workshop Proceedings. http://ceur-ws.org/Vol-1695/paper4.pdf
17. Nickel M, Murphy K, Tresp V, Gabrilovich E (2015) A Review of Relational Machine Learning for Knowledge Graphs. Proc IEEE 104(1):11–33. https://doi.org/10.1109/JPROC.2015.2483592,1503.00759
18. Rosen R, Boschert S, Sohr A (2018) Next Generation Digital Twin. atp magazin 60(10):86. https://doi.org/10.17560/atp.v60i10.2371, http://ojs.di-verlag.de/index.php/atp_edition/article/view/2371
19. Gómez-Berbís JM, de Amescua-Seco A (2019) SEDIT: Semantic Digital Twin Based on Industrial IoT Data Management and Knowledge Graphs. pp 178–188. https://doi.org/10.1007/978-3-030-34989-9_14, http://link.springer.com/10.1007/978-3-030-34989-9_14
20. Dai S, Zhao G, Yu Y, Zheng P, Bao Q, Wang W (2021) Ontology-based information modeling method for digital twin creation of as-fabricated machining parts. Robot Comput Integr Manuf 72:102173. https://doi.org/10.1016/j.rcim.2021.102173, https://www.sciencedirect.com/science/article/pii/S0736584521000570
21. Banerjee A, Dalal R, Mittal S, Joshi KP (2017) Generating Digital Twin models using Knowledge Graphs for Industrial Production Lines. In: Workshop on Industrial Knowledge Graphs, co-located with the 9th International ACM Web Science Conference 2017. https://doi.org/10.1145/3091478.3162383
22. Arp R, Smith B, Spear AD (2015) Building Ontologies with Basic Formal Ontology, vol 91. The MIT Press, Cambridge. https://doi.org/10.7551/mitpress/9780262527811.001.0001, https://direct.mit.edu/books/book/4044, https://www.cambridge.org/core/product/identifier/CBO9781107415324A009/type/book_part
23. Li Y, Chen J, Hu Z, Zhang H, Lu J, Kiritsis D (2021) Co-simulation of complex engineered systems enabled by a cognitive twin architecture. Int J Prod Res 0(0):1–22. https://doi.org/10.1080/00207543.2021.1971318
24. Meierhofer J, Schweiger L, Lu J, Züst S, West S, Stoll O, Kiritsis D (2021) Digital twin-enabled decision support services in industrial ecosystems. Appl Sci 11(23). https://doi.org/10.3390/app112311418, https://www.mdpi.com/2076-3417/11/23/11418
25. Ochoa JL, Valencia-García R, Perez-Soltero A, Barceló-Valenzuela M (2013) A semantic role labelling-based framework for learning ontologies from Spanish documents. Expert Systems with Applications 40(6):2058–2068. https://doi.org/10.1016/j.eswa.2012.10.017, https://linkinghub.elsevier.com/retrieve/pii/S0957417412011311
26. Adl AE (2016) The cognitive digital twins: Vision, architecture framework and categories. https://www.slideshare.net/slideshow/embed_code/key/JB60Xqcn7QVyjb
27. Abburu S, Berre AJ, Jacoby M, Roman D, Stojanovic L, Stojanovic N (2020) Cognitwin–hybrid and cognitive digital twins for the process industry. In: 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), IEEE, pp 1–8. https://doi.org/10.1109/ICE/ITMC49519.2020.9198403
28. Lu J, Zheng X, Gharaei A, Kalaboukas K, Kiritsis D (2020b) Cognitive twins for supporting decision-makings of internet of things systems. In: Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing, Springer, pp 105–115. https://doi.org/10.1007/978-3-030-46212-3_7
29. Albayrak Ö, Ünal P (2020) Smart steel pipe production plant via cognitive digital twins: A case study on digitalization of spiral welded pipe machinery. In: Cybersecurity workshop by European Steel Technology Platform, Springer, pp 132–143
30. Rožanec JM, Lu J, Rupnik J, Škrjanc M, Mladenić D, Fortuna B, Zheng X, Kiritsis D (2021) Actionable cognitive twins for decision making in manufacturing. International Journal of Production Research pp 1–27. https://doi.org/10.1080/00207543.2021.2002967
31. Zhou G, Zhang C, Li Z, Ding K, Wang C (2020) Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing. Int J Protein Res 58(4):1034–1051
32. Cloutier R, Sauser B, Bone M, Taylor A (2014) Transitioning systems thinking to model-based systems engineering: Systemigrams to sysml models. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45(4), 662–674, DOI: 10.1109/TSMC.2014.2379657
33. Haskins C (2014) A Journey Through The Systems Landscape. INSIGHT 17(2):63–64. https://doi.org/10.1002/inst.201417263a, http://doi.wiley.com/10.1002/inst.201417263a
34. Gharaei A, Lu J, Stoll O, Zheng X, West S, Kiritsis D (2020) Systems engineering approach to identify requirements for digital twins development. In: Lalic B, Majstorovic V, Marjanovic U, von Cieminski G, Romero D (eds) Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems, Springer International Publishing, Cham, pp 82–90. https://doi.org/10.1007/978-3-030-57993-7_10
35. Zheng X, Lu J, Kiritsis D (2021) The emergence of cognitive digital twin: vision, challenges and opportunities. International Journal of Production Research pp 1–23
36. Lu J, Zheng X, Schweiger L, Kiritsis D (2021) A cognitive approach to manage the complexity of digital twin systems pp 105–115
37. Botta A, de Donato W, Persico V, Pescapé A (2016) Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems 56:684–700. https://doi.org/10.1016/j.future.2015.09.021, https://linkinghub.elsevier.com/retrieve/pii/S0167739X15003015
38. Minerva R, Biru A, Rotondi D (2015) Towards a definition of the Internet of Things (IoT). IEEE Internet Initiative. https://doi.org/10.1111/j.1440-1819.2006.01473.x
39. Lu J, Chen D, Wang G, Kiritsis D, Törngren M (2021) Model-based systems engineering tool-chain for automated parameter value selection. IEEE Transactions on Systems, Man, and Cybernetics: Systems pp 1–15. https://doi.org/10.1109/TSMC.2020.3048821

40. Lomov I, Lyubimov M, Makarov I, Zhukov LE (2021) Fault detection in tennessee eastman process with temporal deep learning models. J Ind Inf Integr 23:100216. https://doi.org/10.1016/j.jii.2021.100216, https://www.sciencedirect.com/science/article/pii/S2452414X21000145
41. Chen J, Hu Z, Lu J, Zhang H, Huang S, Torngren M (2019) An open source lifecycle collaboration approach supporting internet of things system development. In: 2019 14th Annual Conference System of Systems Engineering, SoSE 2019. https://doi.org/10.1109/SYSOSE.2019.8753883
42. Lu J (2016) A Model-driven and Tool-integration Framework for Whole Vehicle Co-simulation Environments. 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016) https://hal.archives-ouvertes.fr/hal-01280473/
43. Lu J, Wang G, Ma J, Kiritsis D, Zhang H, Törngren M (2020) General Modeling Language to Support Model-based Systems Engineering Formalisms (Part 1). INCOSE International Symposium 30(1):323–338. https://doi.org/10.1002/j.2334-5837.2020.00725.x, https://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2020.00725.x
44. Guo J, Wang G, Lu J, Ma J, Törngren M (2020) General Modeling Language Supporting Model Transformations of MBSE (Part 2). INCOSE International Symposium 30(1):1460–1473. https://doi.org/10.1002/j.2334-5837.2020.00797.x, https://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2020.00797.x
45. Lu J, Wang J, Chen D, Wang J, Törngren M (2018) A service-oriented tool-chain for model-based systems engineering of aero-engines. IEEE Access 6:50443–50458. https://doi.org/10.1109/ACCESS.2018.2868055
46. Lu J, Ma J, Zheng X, Wang G, Li H, Kiritsis D (2021) Design ontology supporting model-based systems engineering formalisms. pp 1–12. https://doi.org/10.1109/JSYST.2021.3106195
47. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Thiel K, Wiswedel B (2009) KNIME - the Konstanz information miner. ACM SIGKDD Explorations Newsletter 11(1):26–31. https://doi.org/10.1145/1656274.1656280, http://portal.acm.org/citation.cfm?doid=1656274.1656280, https://dl.acm.org/doi/10.1145/1656274.1656280
48. O'Connor M, Das A (2009) SQWRL: A query language for OWL. In: CEUR Workshop Proceedings. http://webont.org/owled/2009/papers/owled2009_submission_42.pdf
49. Masolo C, Borgo S, Gangemi A, Guarino N, Oltramari A, Schneider L (2003) Dolce: a descriptive ontology for linguistic and cognitive engineering. WonderWeb Project, Deliverable D17 v2 1:75–105. https://doi.org/10.3233/AO-210259