Cognitive Thread Supports System of Systems for Complex System Development

Shouxuan Wu
School of Mechnical Engineering
Beijing Institute of Technology
Beijing, China

Jinzhi Lu*
EPFL SCI-STI-DK, Station 9,
CH-1015 Lausanne,
Switzerland
jinzhi.lu@epfl.ch

Zhenchao Hu
School of Mechnical Engineering
Shanghai Jiaotong University
Shanghai, China

Pengfei Yang
Beijing Zhongke Fengchao LtD
Beijing, China

Guoxin Wang*
School of Mechnical Engineering
Beijing Institute of Technology
Beijing, China
wangguoxin@bit.edu.cn

Dimitris Kiritsis
EPFL SCI-STI-DK, Station 9,
CH-1015 Lausanne,
Switzerland

2021 16th International Conference of System of Systems Engineering (SoSE) | 978-1-6654-4454-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/SOSE52739.2021.9497473

***Abstract*—Model-based Systems Engineering (MBSE) has been widely used in the development of complex systems. The system architectures, organizations, research and development processes using MBSE to design complex systems can be seen as a System of Systems (SoS), which has high complexity and hard to manage. The concept of digital thread is proposed to integrate all the models and data in the SoS. However, lack of cognition ability makes it hard to connect the models and data with human, processes and things in the SoS, which reduces the efficiency of complex system development. In this paper, a new concept named Cognitive Thread is first proposed as digital thread with augmented semantic capabilities for identifying the information of the SoS. Then a cognitive thread construction approach based on Open Services for Lifecycle Collaboration (OSLC) specification and knowledge graphs is proposed to support decision-making and management in the SoS. Finally, the feasibility of the proposed approach is verified through a case study of the advanced driver-assistance system development.**

Keywords—Digital Thread, Cognitive Thread, OSLC specification, System of Systems, Model-based Systems Engineering

I. INTRODUCTION

The engineered systems today are becoming increasingly complex due to numerous reasons like increasing system scale. For example, more than 1000 coupling electronic control units are integrated in an avionics subsystem of a commercial aircraft [1]. In recent years, the International Council on Systems Engineering has proposed the concept of Model-based Systems Engineering (MBSE) [3], which uses models to formalize system characteristics during the entire system lifecycle and support activities like requirement analysis, design, verification, operation and maintenance activities starting from the conceptual design stage. By using MBSE, the constituents of complex system are operated and managed independently. The geographic distribution of constituents of complex system also brings about emergent behavior and evolutionary development processes. System that gathering all the system architectures, organizations, research and development (R&D) processes to design complex systems can be seen as a typical System of Systems (SoS) [2].

However, it still exists some challenges in the SoS using MBSE: Models from different domains have heterogeneous data structures and modeling rules, which are difficult to integrate. It is necessary to realize the unified expression and interoperation of heterogeneous models through a unified formalization method. Besides, the strong dependencies across the domains and lifecycle phases between different models are too complex to manage. So the cross-domain dependencies between processes, systems and models need to be captured and defined in a unified way. Moreover, to improve the development efficiency, the models need to be combined with the development activities (traceability management, change management, etc.) in the SoS.

Digital twins and digital thread based on MBSE are considered as solutions to parts of the above challenges [4]. MBSE formally expresses the system through the system models, taking system models as the Authoritative Source of Truth (AST) in the SoS. Digital twins are developed by system models in MBSE to describe the system characteristics, behaviors, process and performance of physical entities [5]. Digital thread is a technical concept frame based on AST, providing the ability of linking digital twins to access and integrate the data, the information and the knowledge of the SoS [6]. Although digital thread based on MBSE can describe different views of the SoS by modeling, it still lacks of the cognition ability for analyzing these information for supporting development activities in the SoS [7], such as extracting historical design knowledge for decision making, predicting the purpose of designers and providing modeling supports by their modeling behaviors, etc. Therefore, it is hard to automatically integrate models, human, processes and things in the SoS to improve the R&D efficiency.

To address the problems above, we proposed a concept frame and an approach to support the SoS in this paper. We first propose a new concept named Cognitive Thread (CT) especially for industrial enterprises that developing complex products. Then based on the proposed concept of CT and knowledge graphs (KGs) modeling, a CT construction approach is proposed. Through the KGs, the CT can identify and manage the interrelationship of information in the SoS and realize the end-to-end traceability management by reasoning. Finally, the feasibility and effectiveness of the proposed approach is verified through a case study of the advanced driver-assistance system design.

The rest of the paper is organized as followed. We discuss related work in Section II and introduce the CT definition in Section III. Moreover, a CT construction approach based on KGs is proposed in Section III. Section IV introduces a case study about the CT construction for the advanced driver-assistance system to verify our approach. Finally, we discuss the case study in Section IV and offer the conclusions in Section V.

II. Related Work

The development of system modeling and simulation technology promotes the generation and development of the concept of digital thread. Initially, the digital thread has been successfully applied to the design of the next generation of tankers and F-22 fighter as an extensible and configurable enterprise analysis framework [8], providing model-based analysis and evaluation for the system, network, architecture in the early design stage. The usage of digital thread effectively reduces the number of the wind tunnel test and flight test.

Since then, the digital thread has been widely used to support the integration and interoperation of the heterogeneous data in the engineering product lifecycle. On the one hand, semantic technology provides reasoning and query capabilities. Bone *et al.* [9] proposed an interoperability and integration framework based on ontology and provided the ability of design information retrieval based on semantic model and ontology modeling language. On the other hand, a large number of open standards have emerged to support integration and interoperation, such as the Open Services for Lifecycle Collaboration (OSLC) specification[10] and the Functional Mock-up Interface (FMI) standard [11]. OSLC provides common service definitions for communicating information and a set of common technical protocols for daily working methods and lightly couple automation (e.g., service discovery, query, delegated user interface, change logs) to support lifecycle collaboration. Syndeia[1], Smartfact[2], and the OSLC code generator Lyo-Designer [12] are some of the software that support OSLC specification. Besides, Organization for the Advancement of Structured Information Standards (OASIS) proposed OSLC core to specify the common features that every OSLC Service can be expected to support [13].

In research fields, Lu *et al.* [14]proposed a service-oriented tool-chain framework, constructing the domain specific models of aero engine based on the GOPPRR meta-meta-model modeling method. By using OSLC specification, a model-driven DT platform was built up from the perspective of process, which also realized the co-simulation between function mock-up units (FMUs) and Simulink models. Gray *et al.* [15] using System Modeling Language (SysML) to define the requirements, specification, functions, physical operations and physical performance of regional aircraft de-icing system in CRYSTAL project, using OSLC specification and FMI standard to realize the traceability management of requirements, change requests and tests as well as heterogeneous simulation in the lifecycle.

The digital thread consists of a large number of lifecycle models and data with topological relationships. In recent years, KGs has been widely used in the construction and management of digital Thread as an enabling technology which links models and data [16]. Kwon *et al.* [17] and Helu *et al.* [18]described the model and data in manufacturing and inspection stage of the mechanical product lifecycle based on Standard for The Exchange of Product model data (STEP) and the Quality Information Framework (QIF) standard. Through establishing the semantic mapping between EXPRESS/XML format data and ontology, the model and the data were transformed into KGs and semantic query based on SWRL was implemented to achieve decision-making through reasoning. Hedberg *et al* [19] extended the method of linking data in manufacturing context in distributed manufacturing system and built a handler system based on Distributed Object Identifier (DOI) and KGs, supporting the creation, track and query of the association between digital artifacts established based on specifications and standards such as SysML, XML, STEP and QIF.

In conclusion, there exist opportunities that brought by combining the digital thread, open standards and the KGs. Besides, considering the possibility of combining MBSE with artificial intelligence technology in the future [20], the implement of combining the mentioned technologies can realize the vision of smart system development such as cognitive decision-making, intelligent-based data integration and visualization.

III. Cognitive Thread

A. Basic Concept

In this section, basic concepts of *Digital Thread* and *Cognitive Thread* (CT) are introduced. Based on these concepts, the differences between them are summarized.

In the SoS, the basic concept of *Digital Thread* is defined as follows:

Digital Thread: *Digital Thread is a technical concept frame based on authoritative source of truth, providing the ability of linking digital twins to access and integrate the data, the information and the knowledge of the SoS.* [6]

Compared with the integration ability that *Digital Thread* emphasizes, the CT emphasizes the cognitive ability from *Cognitive Twins* (CTw) [21] for analyzing the integrated information and giving feedback. In the SoS, the basic concept of CT is defined as follows:

Cognitive Thread(CT): *Cognitive Thread is a technical concept frame to identify the interrelationships of data, information and knowledge in the SoS, as well as the process of understanding and analyzing these interrelationships for supporting development activities.*

The technical concept frame of CT is proposed especially for industrial enterprises that developing complex products. The CT extends the characteristics of *Digital Thread* to integrate and link the elements in the SoS and the virtual SoS. Therefore, the CT can describe the information of the SoS in different levels of granularity like SoS, system and component. Moreover, development scenarios are created based on the use cases in the specific lifecycle phases during the development process. The CT also extends the cognitive characteristics of CTw to recognize the scenarios in development process. In addition, based on the cross-domain dependencies between the identified information, the CT can implement adaptive reasoning and decision-making to support activities in the SoS.

As shown in Fig. 1, the SoS of a commercial aircraft consists of elements like physical assets (aircraft engine and undercarriage, etc.), organizations (managers and engineers, etc.), scenarios (modeling, simulation and testing, etc.), development activities (traceability management, consistency management, etc.), development processes (requirement analysis, preliminary design phase, detailed design and V&V, etc.), digital systems (composed by standardization requirements documents, system architecture models,

[1] Syndeia (Intercax): http://intercax.com/products/syndeia/

[2] Smartfact: https://smartfacts.com/

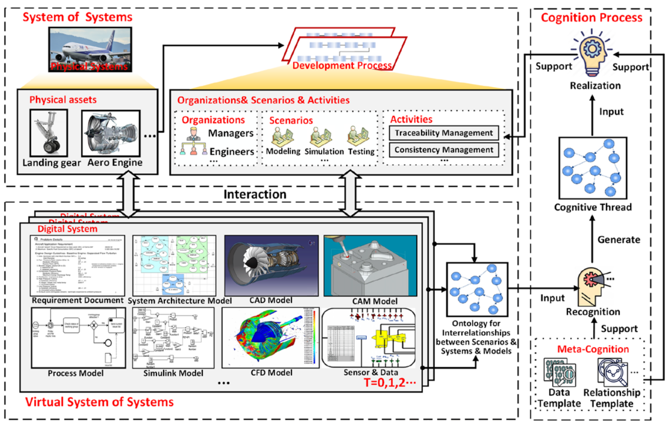


Fig.1. Concept of CT

Simulink and CFD models, CAD and CAM models, sensor and data, process models etc.) and the interactions of these elements.

To identify the elements mentioned above, we adopt the concept of cognition [22] in psychology and extend the concept by refining the cognition. *The cognition is the process of acquiring, integrating, transforming information of the SoS and generating CT for supporting development activities. The meta-cognition is a collection of templates determining how cognition process works*. For example, in traceability management, The meta-cognition provides templates about how to identify the key elements of traceability relationships and how to manage the traced relationships. Based on these templates in meta-cognition, the ontology which represents key elements (such as modeling scenarios and models, etc.) of traceability relationships is captured to generated CT. Then the CT provides information about the traced relationships based on the information of ontology. The engineers can use the information to make decision like tracing specific data sources or managing the traceability.

We define the cognition process by the process of recognition and realization, separately, as shown in Fig. 1. *Recognition is the process of accepting and understanding the characteristics of elements and their interrelationships in the SoS.* Based on templates of meta-cognition, recognition identifies the ontology which describes interrelationships between scenarios, systems and models in the SoS and generates CT. *Realization is the process of encoding, storing and extracting the information of elements and their interrelationships in the SoS for supporting development activities.* Based on templates in meta-cognition, the realization identifies the information in generated CT and uses these information for decisions-making or management in specific development activities.

The formalized concept definition of CT is as followed:

$$CT_{SoS}=P_{SoS} \cup V_{SoS} \{ \Sigma D_{sys}(\Sigma Model\ (Mc,Mp,Mm,Ml,Mt)\ ), OntologyRel \} \cup Interaction\{ \Sigma Oper\ (O_c,O_r,O_u,O_d), \Sigma Data\ (D_d, D_p,D_c) \} \cup Cog\{ Meta\text{-}Cog(\Sigma Temp), \Sigma Scenarios(S_s, S_s,S_d), \Sigma Realization(R_p,R_m,R_t,R_t) \}$$

Where CT_{SoS} refers to the CT of the SoS. P_{SoS} refers to the physical existence of the SoS, including organizations, scenarios, activities, processes and physical assets. V_{SoS} refers to the virtual representation of the SoS. V_{SoS} is consisted by a collection of digital systems D_{sys} and the *ontologyRel*. D_{sys} refers to the digital system presenting information related to the SoS . D_{sys} is consisted by a collection of models. Each *Model* includes several features as followed:

- M_c *(Model Content)*: The information contained in the model, including topology structure, parameter attributes and other information of the model.
- M_p *(Modeling Purpose)*: The view of modeling, representing the value of the model.
- M_l *(Modeling Language)*: The language for presenting information, knowledge and system in a specific rule or approach.
- M_m *(Modeling Method)*: The method defining how to use modeling language to achieve modeling purpose.
- M_t *(Modeling Theory)*: The theoretical foundations of modeling, such as mathematical algorithms and formalized expressions that describe systems.
- M_t *(Modeling Tool)*: The tool for establishing model and implementing specific functions based on models.

We used *interaction* to describe how the information of the SoS is organized. *Interaction* includes a collection of *Oper* and *Data*. *Oper* refers to operations or behaviors that change the information in the SoS. O_c ,O_r ,O_u and O_d in Each *oper* refer to the creation, retrieval, update and deletion of information. *Data* refers to the data transmission and transformation in the SoS. Each *Data* includes several features as followed:

- D_d *(Data flow Direction)*: The direction of data interaction, including the beginning and the end of the data interaction .
- D_p *(Data Property)*: The properties of data interaction, including data type, data standards and interfaces.
- D_c *(Data Content)*: The content value for each data interaction.

Cog and *Meta-Cog* refer to the cognition process and meta-cognition. *Meta-Cog* is consisted by a collection of templates *temp* for defining how cognition processes work. *Cog* includes a collection of *scenarios* of recognition and *realization* processes. During cognition process, scenarios are specific use cases in the development process. Each *scenario* includes several features as followed:

- S_s *(Scenario Scope)*: The purpose of the constructed scenario, such as requirements analysis, system architecture design, detailed design and other purposes.
- S_s *(Scenario Structure)*: The description of the scenario, such as the system engineers use SysML modeling tool for architecture designing.
- S_d *(Scenario Domain)*: The domain description of the scenario. For example, a scenario can be classified into mechanical, control, electronics, etc.

In *Realization* process, the information of the SoS and templates in meta-cognition are used in the behaviors like reasoning. These behaviors support decision-making in

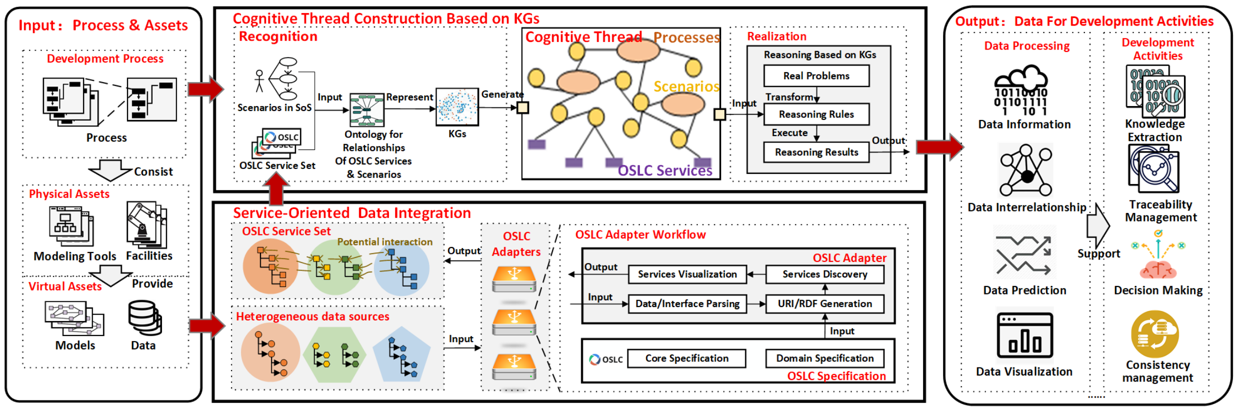


Fig.2. Workflow of the CT Construction Approach

specific development scenarios or processes. Each *realization* includes several features as followed:

- R_p *(Realization Purpose)*: The goal needs to be achieved, such as tracing the requirement items.
- R_m *(Realization Method)*: The method defining how to achieve realization purpose, such as rules of reasoning.
- R_t *(Realization Theory)*: The abstract representation of realization method, such as mathematical model based on algorithms.
- R_t *(Realization Tool)*: The tool for implementing realization method, such as reasoner.

B. *A Cognitive Thread Construction Approach*

In order to construct CT for complex product R&D lifecycle, a CT construction approach based on OSLC specification and KGs is proposed as shown in Fig. 2. It requires inputs from process and assets (such as modeling tools, facilities, models and data, etc.) and provides outputs of data for development activities (such as extraction of historical design knowledge, traceability management of development information, decision making and consistency management of development information, etc.). The two main composition of the approach are shown in details as follows:

Service-Oriented Data Integration: As different modeling tools provide heterogeneous data sources. end-to-end integration of tools can only be realized by developing customized tool interfaces. Therefore the integration and interoperability among tools in large-scale tool-chain are limited. The OSLC specification provides a loosely coupled foundation for domain specific tool integration. The OSLC adapters are developed to realize the transformation of heterogeneous data sources to a set of services uniformly represented by the OSLC specification. The OSLC adapters parse the data/interfaces of the input heterogeneous data sources, then generating URI/RDF presented OSLC services based on OSLC core and domain specification. After establishing the logics of discovering OSLC services, the OSLC services are . Finally, models and data from cross-domain tools are transform into an unified OSLC services set which enables the potential data integration and interoperability.

CT Construction Based on KGs: Based on development scenarios of stakeholders in development processes and all the generated OSLC services, ontology is used to describe the interrelationships between OSLC services and scenarios. KGs are used to represent the information of ontology (such as the source and target of OSLC services links as well as the creator, creation time, creation tools of OSLC services, etc.). After the construction of KGs, the CT are constructed to describe the information and interrelationships between development processes, related scenarios and related OSLC services. Moreover, reasoning based on KGs enables to realization based on the constructed CT. Engineering problems in development scenarios are transformed by some problems templates into a URI. This URI includes all the information about the problems and can be transformed into reasoning rules of KGs. Based on these reasoning rules, reasoning executor matches the corresponding information in CT. The relevant information of OSLC services are found as the reasoning results to support the specific development activities.

IV. Case Study

A. Problem Statement

Advanced driver-assistance system (ADAS) is a typical complex electromechanical system that supports an automobile to give emergency braking when a collision risk is detected by the sensor to reduce collision speed and avoid collision . The ADAS R&D lifecycle consists of several phases such as requirement analysis, designing, testing and manufacturing. Since the R&D lifecycle of ADAS is relatively long, a CT is built to realize the integration and traceability management of models and data.

B. *Cognitive Thread Supports Traceability Management*

Based on the approach we proposed in Section. III, we used several domain modeling tools to develop ADAS. As Fig.3.a shows, in the requirements definition phase, *Excel* is adopted to formalize the development requirements. The requirement documents are imported into the multi-domain specific modeling tool *MetaGraph* (available: http://www.zkhoneycomb.com/) to automatically generate requirement models based on SysML. In the conceptual design phase, requirement models are extended to develop the architecture models for ADAS from the views (function, logic and physics, etc.) based on a multi-architecture textual modeling language KARMA [23] and its modeling method of graph, object, point, property, relationship, role and extension (GOPPRRE). The architecture models can be transformed into domain models, documents and other digital assets for verification and more detailed designs through code generation [24], such as Simulink model in *MATLAB* and

CAD model in the *AutoCAD*. In order to transform these models and data mentioned above into OSLC services, we develop OSLC adapters for domain-tools based on baseline of OSLC core specification 2.0 and OSLC adapters develop-tool *Datalinks* (available: http://www.zkhoneycomb.com/). All the data and interfaces are transformed by OSLC adapters into OSLC services represented by URLs automatically. In addition, we develop a tool named *OSLC Service Manager* to manage the generated OSLC services and generate CT. In *OSLC Service Manager,* system engineers can establish links of OSLC services based on development scenarios manually and generated KGs. The KGs are considered as an prototype CT to describe the information and interrelationship of OSLC services and ADAS development scenarios.

As Fig.3.b shows, the real problems (e.g: I want to know all the relevant component information for Simulink model component *Out1* of model named *vdp* in context.) in development activities are transformed by some problems templates into a URI. This URI will be captured by CT and transformed into reasoning rules based on cypher query language. Moreover, we also develop OSLC service plugins of different domain tools to access the reasoning results. For example, we develop an OSLC service *MATLAB*-plugin based on Matlab GUI. The simulation engineer uses this plugin to find reasoning results. By clicking and visiting the URL links in reasoning results, the simulation engineer can get all the information such as the requirement item *REA-A-REQ10* , the CAD model *pushrod*, the *Fcn* module in Simulink model and the block in state machine diagram SysML model. Based on these information, the engineer can making better decision or management in the SoS of designing ADAS.

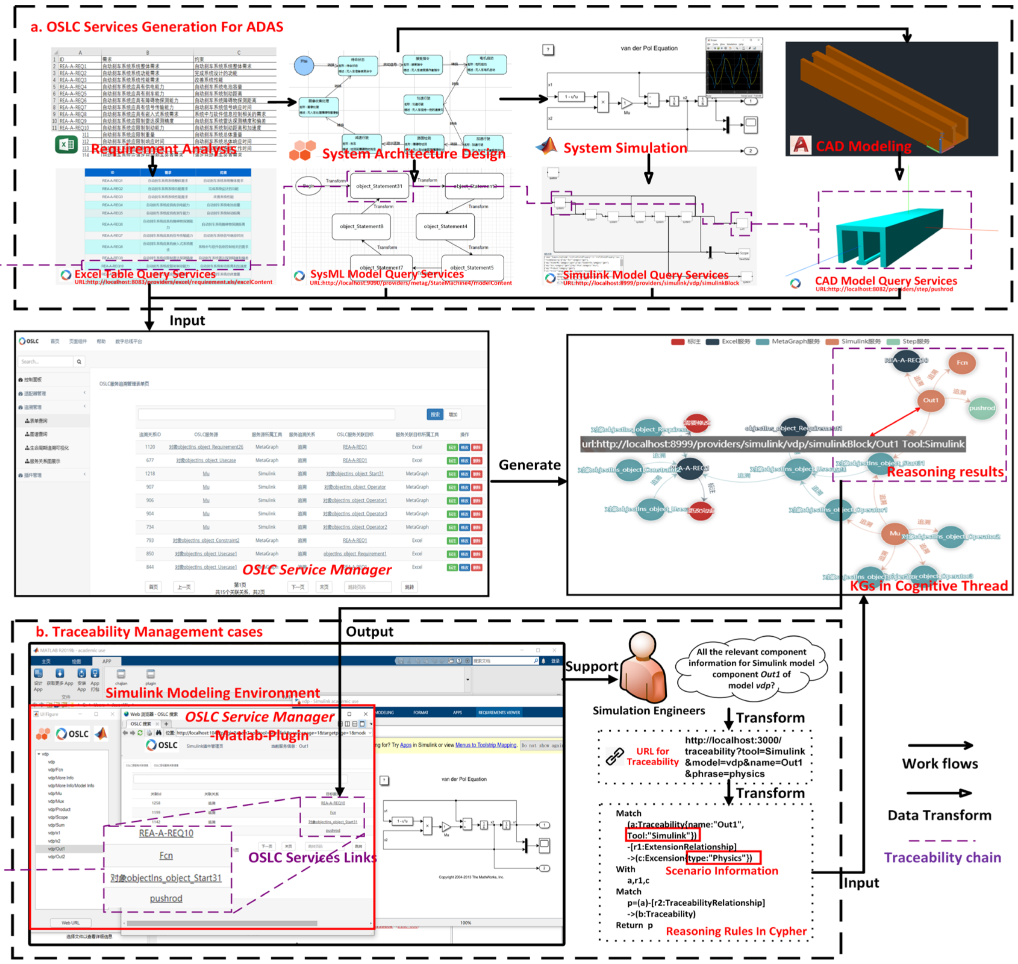
C. Discussion

In case study of ADAS system design, we adopted OSLC specification and KGs to integrate heterogeneous models and construct CT of a unified data source. The constructed CT realize the traceability management of SysML models, Simulink models, CAD models and Excel tables. Compared with ADAS development approach in [25], the superiorities of the proposed approach are as follows:

1) More flexible integration ability: OSLC provides a better lifecycle-range standardized integration capability for tools and interfaces in the future. All the OSLC services are generated by OSLC adapters that provided by tools supplier, which means that only the cost of access data through URLs is needed.

2) Better understanding about the SoS: KGs in CT provide the ability of linking information and reasoning based on the OSLC services and scenarios in the SoS, which promotes the understanding of lifecycle data and data traceability for skateholders (engineers, managers, etc.).

3) Better automation and tool accessibility: The generation processes of OSLC services from heterogeneous


a. OSLC Services Generation For ADAS
Requirement Analysis
System Architecture Design
System Simulation
CAD Modeling
Excel Table Query Services
SysML Model Query Services
Simulink Model Query Services
CAD Model Query Services
Input
Generate
OSLC Service Manager
url:http://localhost:8999/providers/simulink/vdp/simulinkBlock/Out1 Tool:Simulink
REA-A-REQ10
Fcn
Out1
pushrod
Mu
Reasoning results
KGs in Cognitive Thread
b. Traceability Management cases
Output
Simulink Modeling Environment
OSLC Service Manager
Matlab-Plugin
REA-A-REQ10
Fcn
对象objectIns_object_Start31
pushrod
OSLC Services Links
Support
Simulation Engineers
All the relevant component information for Simulink model component Out1 of model vdp?
Transform
URL for Traceability
Transform
Scenario Information
Reasoning Rules In Cypher
Input
Work flows
Data Transform
Traceability chain

```
http://localhost:3000/
traceability?tool=Simulink
&model=vdp&name=Out1
&phrase=physics
```

```
Match
    (a:Traceability{name:"Out1",
    Tool:"Simulink"})
    -[r1:ExtensionRelationship]
    ->(c:Excension{type:"Physics"})
With
    a,r1,c
Match
    p=(a)-[r2:TraceabilityRelationship]
    ->(b:Traceability)
Return  p
```

Fig.3. Traceability Management process of ADAS Braking Subsystem

data sources are almost automatic. Besides, developed OSLC services management tool and plug-ins provide visualized traceability links and information for skateholders, which improves the design efficiency in the SoS.

Though the CT supports traceability management activity in the SoS of ADAS design, several limitations exist: Firstly, the implement of CT concept is now not applicable to all the SoS. In this paper, we only focus on the development system (viewed as a SoS) to develop a complex system. Secondly, we use the cases of ADAS design to verify that our concept and approach are feasible, but the correctness and completeness of formalism for CT are under validating. Finally, different approaches of developing OSLC adapters brings about different granularity of CT, which means that the measurements of the return on investment of building CT are needed.

V. Conclusion

In this paper, we propose a new concept of Cognitive Thread (CT) and a CT construction approach to support the system of systems of complex systems. The OSLC services are used to uniformly represent the elements in the system of systems. KGs are used to link OSLC services and to construct CT for scenario-based reasoning. Finally, a case study of an ADAS system is conducted to verify the proposed approach. In order to improve the concept of CT, we hope to combine the CT with AI technology (such as natural language processing and AR) to realize the intelligent-based cognition in the system of systems.

References

[1] H. Salzwedel, "Mission level design of avionics," *AIAA/IEEE Digit. Avion. Syst. Conf. - Proc.*, vol. 2, pp. 1–10, 2004.

[2] M. W. Maier, "Architecting Principles for Systems-of-Systems," INCOSE Int. Symp., vol. 6, no. 1, pp. 565–573, 1996.

[3] Hart L E. Introduction to model-based system engineering (MBSE) and SysML. Delaware Valley INCOSE Chapter Meeting, Ramblewood Country Club, Mount Laurel, New Jersey.

[4] Kraft, Edward M. "A Disruptive Application of Digital Engineering to Optimize Aircraft Developmental Test & Evaluation". AIAA-2018-2853, 2018 Aviation Systems Conference, Atlanta, GA, June 25-29, 2018.

[5] C.Zhuang, J.Liu, H.Xiong, X.Ding, S.Liu, G.Wen.Connotation, architecture and trend of product digital twin. Jisuanji Jicheng Zhizao Xitong/Computer Integr Manuf Syst CIMS.

[6] T. D. West and A. Pyster, "Untangling the Digital Thread: The Challenge and Promise of Model-Based Engineering in Defense Acquisition," Insight, vol. 18, no. 2, pp. 45–55, 2015.

[7] D. N. Mavris, M. Balchanos, O. J. Pinon, and W. J. Sung, "Towards a digital thread-enabled framework for the analysis and design of intelligent systems," AIAA Inf. Syst. Infotech Aerospace, 2018, no. 209989, 2018.

[8] E. M. Kraft, "HPCMP CREATETM-AV and the air force digital thread," 53rd AIAA Aerosp. Sci. Meet., no. January, pp. 1–13, 2015.

[9] M. Bone, M. Blackburn, B. Kruse, J. Dzielski, T. Hagedorn, and I. Grosse, "Toward an Interoperability and Integration Framework to Enable Digital Thread," Systems, vol. 6, no. 4, p. 46, 2018.

[10] "OSLC–OASIS open services for lifecycle collaboration." [Online].Available: http://www.oasis-oslc.org/.

[11] T. Blochwitz, M. Otter, et al. Functional Mock-up Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In Proceedings of the 9th International Modelica Conference, 2012.

[12] J. El-khoury, "Lyo code generator: A model-based code generator for the development of OSLC-compliant tool interfaces," SoftwareX, vol. 5, pp. 190–194, 2016.

[13] OASIS: Open Services for Lifecycle Collaboration Core Specification Version 3.0 (2017). http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/part1-overview/oslc-core-v3.0-cs01-part1-overview.html. Accessed 21 December 2020

[14] J. Lu, J. Wang, D. Chen, J. Wang, and M. Torngren, "A service-oriented tool-chain for model-based systems engineering of aero-engines," IEEE Access, vol. 6, pp. 50443–50458, 2018.

[15] G. Bachelor, E. Brusa, D. Ferretto, and A. Mitschke, "Model-Based Design of Complex Aeronautical Systems through Digital Twin and Thread Concepts," IEEE Syst. J., vol. 14, no. 2, pp. 1568–1579, 2020.

[16] A. Banerjee, R. Dalal, S. Mittal, and K. P. Joshi, "Generating Digital Twin models using Knowledge Graphs for Industrial Production Lines," Work. Ind. Knowl. Graphs, co-located with 9th Int. ACM Web Sci. Conf. 2017, no. June, pp. 1–5, 2017,

[17] S. Kwon, L. V. Monnier, R. Barbau, and W. Z. Bernstein, "Enriching standards-based digital thread by fusing as-designed and as-inspected data using knowledge graphs," Adv. Eng. Informatics, vol. 46, no. April, p. 101102, 2020.

[18] M. Helu, A. Joseph, and T. Hedberg, "A standards-based approach for linking as-planned to as-fabricated product data," CIRP Ann., vol. 67, no. 1, pp. 487–490, 2018.

[19] T. D. Hedberg, M. Bajaj, and J. A. Camelio, "Using Graphs to Link Data Across the Product Lifecycle for Enabling Smart Manufacturing Digital Thread," J. Comput. Inf. Sci. Eng., vol. 20, no. 1, 2020.

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

[21] J. Lu, X. Zheng, A. Gharaei, K. Kalaboukas, and D. Kiritsis, "Cognitive twins for supporting decision-makings of internet of things systems," *arXiv*, pp. 1–11, 2019.

[22] T. O. Nelson, "Metamemory: A Theoretical Framework and New Findings," *Psychol. Learn. Motiv. - Adv. Res. Theory*, vol. 26, no. C, pp. 125–173, 1990.

[23] J. Lu, G. Wang, J. Ma, D. Kiritsis, H. Zhang, and M. Törngren, "General Modeling Language to Support Model-based Systems Engineering Formalisms (Part 1)," *INCOSE Int. Symp.*, vol. 30, no. 1, pp. 323–338, 2020.

[24] J. Guo, G. Wang, J. Lu, J. Ma, and M. Törngren, "General Modeling Language Supporting Model Transformations of MBSE (Part 2)," *INCOSE Int. Symp.*, vol. 30, no. 1, pp. 1460–1473, 2020.

[25] C. Buchholz, T. Vorsatz, S. Kind, and R. Stark, "SHPbench - A Smart Hybrid Prototyping Based Environment for Early Testing, Verification and (user based) Validation of Advanced Driver Assistant Systems of Cars," Procedia CIRP, vol. 60, pp. 139–144, 2017.