2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) | 979-8-3503-6123-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ETFA61755.2024.10710740

Toward Automating the Composition of Digital Twins within System-of-Systems

Milapji Singh Gill*, Jingxi Zhang†, Andreas Wortmann†, Alexander Fay‡
*Institute of Automation Technology
Helmut Schmidt University Hamburg, Germany
milapji.gill@hsu-hh.de
†Institute of Control Engineering of Machine Tools and Manufacturing Units
University of Stuttgart, Germany
{jingxi.zhang, andreas.wortmann}@isw.uni-stuttgart.de
‡Chair of Automation
Ruhr University Bochum, Germany
alexander.fay@rub.de

***Abstract*—Cyber-Physical Production Systems necessitate efficient configuration and continuous reconfiguration to adapt to evolving requirements and shifting goals. This process involves integrating various system structures to derive new functions and behaviors. The concept of the Digital Twin, which allows for effective integration and testing, is instrumental in facilitating this task. However, a fundamental prerequisite is that Digital Twins of individual system components must be composed efficiently and in accordance with dynamic System-of-Systems. We address this challenge by introducing an approach for the automated horizontal and vertical composition of Digital Twins, aimed at minimizing manual intervention and enhancing adaptability. Therefore, a pipeline specifically designed for this goal is proposed, which includes the generation of new functions and behaviors. This approach is intended to provide a foundation for future research.**

***Index Terms*—Digital Twins, System-of-Systems, Composition, Integration, Cyber-Physical Systems**

I. Introduction

Rapidly changing customer requirements and production goals increase the demand for modular and interoperable production systems. In this regard, the System-of-Systems (SoS) approach offers significant potential for flexible and adaptable production by incorporating various Cyber-Physical Systems (CPS) [1], [2]. In this framework, CPSs must undergo both horizontal and vertical composition with other CPSs throughout their life cycle, enabling the development of new functions not possible with standalone systems (see. Fig. 1) [2]. Horizontally, this composition involves linking systems across different process steps during operation, thereby facilitating the exchange of materials, energy, and information within manufacturing networks [3]. Vertically, CPS composition encompasses the integration of systems at various hierarchical levels within engineering, from field devices up to the enterprise level (e.g., according to the Reference Architecture for Industry 4.0 (RAMI 4.0)) [4].

In this context, Digital Twins (DTs), serving as ”virtual representations” of physical assets, play a vital role in CPSs. DTs manage and integrate data and digital models across the asset life cycle, enabling essential digital services [5]. They support

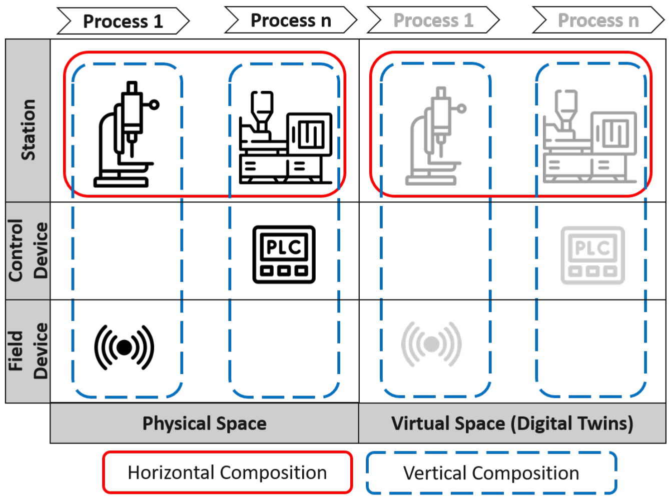

Fig. 1. Horizontal and vertical composition of DTs

bidirectional data exchange for real-time asset monitoring and actions in virtual space to adjust the behavior [6]. Moreover, DTs support varied applications, from virtual commissioning to predictive maintenance [7]. Their role is also critical in the automated composition of systems, from individual CPS to comprehensive Cyber-Physical Production Systems (CPPS) [8], [9]. However, for the effective deployment, DTs must be efficiently synchronized with the corresponding composed SoS. Currently, a high effort and manual intervention is needed for this task. To increase efficiency, more automation is required to integrate evolving system structures and develop composed DT functions and behaviors.

Thus, an outline of a pipeline for the automated composition of DTs in the context of SoS is sketched in this contribution. Sec. II introduces key requirements for this goal, followed by a description of the pipeline in Sec. III. In Sec. IV, related approaches are analyzed with regard to specific pipeline constituents. Finally, Sec. V summarizes the presented contribution and highlights directions for future work.

II. Requirements

R1: Domain-Specific Semantic and Standard-Based Expression of DT Information

The first requirement focuses on integrating and ensuring interoperability among DT components, such as data, digital models, and digital services. The varied origins of DTs present challenges in harmonizing their structures, functions, and behaviors, complicating automated composition. To address this, a formal semantic model is necessary, based on domain-specific standards, that clearly and unambiguously describes all relevant DT information [10], [11]. This semantic model alongside a meta model, which includes information of the targeted SoS, is essential for facilitating further composition and ensuring consistent integration across the pipeline.

R2: Parsing of DT Information for Composition

The composition of DTs relies on seamless interoperability of DT components. DTs contain data and digital models from different domains, necessitating cohesive meta models for input artifacts that integrate diverse domain expertise and distributed data [10]. Leveraging domain-specific languages (DSLs) allows domain experts to contribute effectively, using their preferred modeling methods [12]. These DSLs are commonly designed in language workbenches (LWBs) that automatically provide an infrastructure for model parsing and interpretation.

R3: Automatic Generation of Composed DT Functions

In CPPS, CPSs are often composed both horizontally and vertically, requiring efficient, automated generation of corresponding composed DT functions [2]. To ensure consistency, correctness, and interoperability across abstraction layers, it is essential to use a formal representation of DT components [13]. These formal models enable a systematic synthesis of data and logical inference of new DT functions, facilitating the integration and inference of higher-level DT functions using domain-specific knowledge [14]. This ensures reliable and accurate automatic generation of new DT functions, crucial for the effective operation of CPPS.

R4: Automatic Generation of Composed DT Behavior

Another key pipeline constituent is the automated generation of DT behavior according to newly composed DT functions. Effective automation requires addressing challenges in generating diverse behavior models from these functions. These include maintaining precision in behavior rules, ensuring flexibility for dynamic changes, achieving interoperability across CPS components, and allowing domain experts to influence behavior specifications easily. Additionally, automated code generation is necessary to efficiently translate these behavior models into executable code. Addressing these issues is crucial for maintaining the consistency, correctness, and adaptability of DT behavior. [3]

R5: Automatic Validation and Verification

Automated validation and verification of DTs must resolve component conflicts to ensure specifications and CPS integrity. It is crucial to confirm that composed DTs function as intended and implement their behaviors correctly in real-world contexts. Semantic soundness validation is essential, requiring checks against domain-specific rules and objectives [15]. This includes assessing how well DTs meet targets, such as providing a comprehensive manufacturing system overview as key performance indicators. Finally, detailed reports must document the process's effectiveness, offering insights into the performance as well as reliability of composed DTs.

III. Pipeline for the automatic composition of Digital Twins

A. Overview

We propose the following pipeline in Fig. 2 with different engines for the composition of DTs to accomplish an automated integration. The input includes the *semantic DT model*, supplemented by formal artifacts such as rules and constraints, and meta models for specific pipeline constituents. These are parsed into an abstract syntax tree (AST) containing all semantic DT information required for composition *(1. Parser Engine)*. From this AST, the *semantic DT model*, which includes the formal description of functions, is used together with the meta model as well es formal rules for inferring new functions with reasoning mechanisms and a rule engine *(2. Composition Engine)*. Naturally, the automatic composition can result in unwanted functions and safety risks. Therefore, formal validation is necessary to comply with rapidly changing safety regulations *(3. Validation Engine)*[1]. Finally these new composed DT functions are used for code (behavior model) and report generation *(4. Generator Engine)*.

B. Semantic Digital Twin Model

Given the diversity of DTs from various manufacturers and the heterogeneity of their data sources, a unified *semantic DT model* employing domain-specific standardized terminology and relationships is crucial for integrating data, digital models, and digital services throughout the pipeline. The use of formal models like ontologies provides machine-readable descriptions vital for automated integration and enables reasoning mechanisms in the *Composition Engine* to accurately infer new functions from updated structures. Selecting a top-level ontology relevant to the specific domain is essential for effectively integrating and ensuring the interoperability of DTs. This ontology serves as the foundational semantic framework for the composition process. Thus, a comprehensive *semantic DT model* must encompass details about both the DT and its physical counterpart. Additionally, specifying the technical interfaces necessary for function execution is essential for operation. Beyond the content of the *semantic DT model*, the use of modular and standards-based ontology design patterns, tailored to specific information needs and use cases, is vital for continuous adaptation of the *semantic DT model* [11]. These design patterns provide standardized templates that experts can employ to describe relevant aspects of the DT, including its functions, structures, and behaviors. For the integrity and validity of the composed DTs, providing validation rules and

[1]OSHA safety regulation standards: https://www.osha.gov/laws-regs/interlinking/standards/1910.212

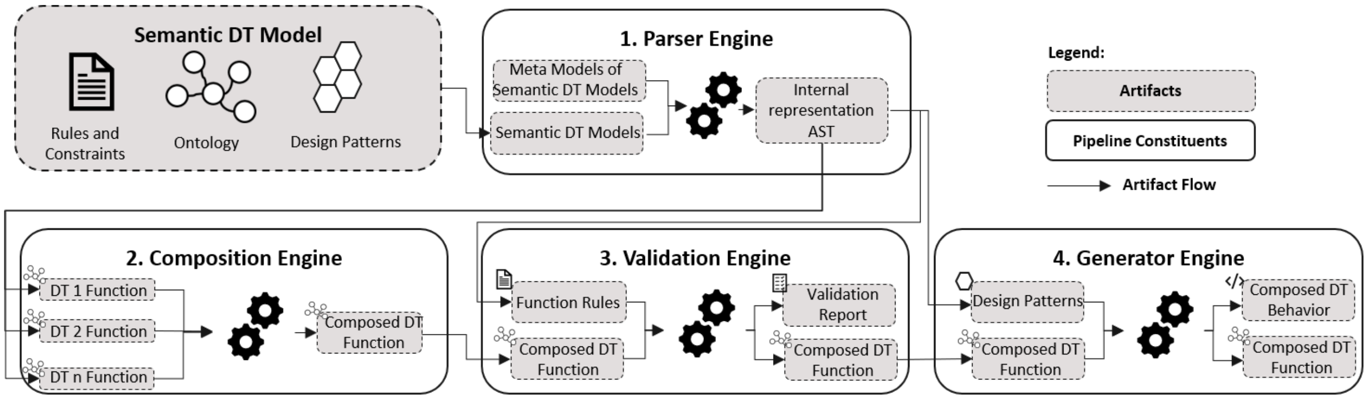

Fig. 2. Pipeline for the composition of DT functions, divided into four core steps.

constraints to the *Validation Engine* is advantageous. In this context, the Shapes Constraint Language (SHACL) can be utilized for data structure validation in order to ensure adherence to the defined *semantic DT model*. Furthermore, applying the Semantic Web Rule Language (SWRL) can enforce complex rules (e.g. regulatory rules), ensuring consistent and correct functions across the composed DTs.

C. Parser and Composition Engine

For effective DT function composition, *semantic DT models* are parsed into an AST for internal representation and reuse. These models, along with the semantic DT function descriptions, are utilized in the following process. When considering DT functions and interfaces of DTs, it's necessary for functions required by one DT and provided by another to conform to a specific schema. Each DT function, tied to both structure and behavior models, must define clear pre- and post-conditions related to its potential behavior. Based on defined structure models with an underlying meta model, models can be composed by integrating one structure with another. Behavior models must be adapted to conform to both the structure and the meta model. Importantly, behavior models must also adhere to a meta model. For instance, structural integrity can be leveraged to infer new structures, and defined behaviors can predict all potential behaviors. Using an ontology of behavior and structure combinations, models can be composed with reasoning mechanisms and a rule engine. Syntactic correctness is ensured as the new composed DT function adheres to predefined syntax, while semantic correctness is verified by validating the composed function models. In adaptation to previous research [3], the composition of Business Process Model Notation (BPMN) for modelling the behavior alongside of class diagrams as a structural component can be treated as a horizontal composition. While an abstraction in a vertical direction involves creating an overview of the system as a whole. For instance, a horizontal composition involves integrating a machine tool with a multi-axis robot, while a vertical composition integrates this system into a product assembly line. As the DT functions are related to their behaviors, the latter are derived from the former. Overall a composed DT function will provide a composition of these pre- and post-conditions. These conditions can be considered for validation against validation rules.

D. Validation and Generator Engine

Other than the DT functions, function rules can be derived from requirements documents or functionality regulations. Using a language such as SWRL, these regulations can be formalized for validation as function rules. This enables test engineers to verify composed DT functions against constraints and regulations within their preferred domain. For instance, a robotic arm's degree of freedom in a product line might be restricted to avoid damaging nearby infrastructure due to space constraints. Test engineers design simulations to test composed functions, ensuring semantic soundness by defining function rules, such as preventing contradictory control commands in machine tool behaviors. The validation strategy involves checking the generated composed DT function against function rules. Specifically, the structure and behavior must be tested against limitations and goals, for example, collision detection in a simulation using 3D CAD models. Output artifacts of this step are the validation report and the composed DT function, which is passed to the *Generator Engine*. Using design patterns, these functions are used to generate the code to provide the composed DT behavior to the physical machine. These templates are required for code generation, e.g. Java code or a CNC program.

IV. Related Approaches

In the context of semantic DT models, the German Industry 4.0 initiative has pioneered setting DT description standards through Asset Administration Shells (AAS) available in various serializations like XML and JSON [10]. Bader and Maleshkova [13] advanced this with a formally structured, RDF-based AAS, suitable for semantic DT models. However, for thorough horizontal and vertical DT integration, the metamodel requires additional refinements. The digital shadow metamodel by Becker et al. [16] and the ontology development method by Hildebrandt et al. [11] offer robust frameworks for expanding these models, focusing on modular, reusable, and standards-based ontology design patterns for detailed DT descriptions.

With regard to the Composition Engine, one method for composing functions involves using an ontology, as demonstrated by Elhabbash et al. [17], who integrated four key aspects (user, environment, platform, and services) into an OWL ontology termed holons. Composing such holons necessitates defining the required and provided interfaces. However, their work does not address the nuances of horizontal or vertical composition of functions. In contrast, a model-based approach by another study [3] facilitates the development of system functions by modeling machine functions with gaps filled by domain experts using tools like the BPMN. This approach leads to an executable BPMN that represents a form of horizontal composition but lacks a vertical perspective. Another study [14] focused on a modeling technique for 3D-CAD information within an ontology, achieving a vertical composition by integrating kinematic functions with geometric descriptions. Yet, this approach does not encompass horizontal composition or a broader model-based strategy. LWBs, such as Xtext [18], provide an infrastructure for compositional development and formal validation of domain-specific languages. Recent enhancements in LWBs support design time validation of domain-specific languages [12], using rules to ensure component semantics within specific domains. Further, validation scenarios defined by Soernig et al. [15] are used to conform models to metamodels using DSL derived rules, enhancing DT function validation.

V. Summary and Future Work

This paper presented a pipeline designed for the automated composition of DTs within SoS. It addresses both vertical and horizontal DT composition by leveraging formal artifacts (e.g. semantic DT model) and using DSLs. The combination of both is vital within the pipeline. Firstly, the semantic DT model is essential for the Composition Engine to develop new DT functions. Secondly, the output from this step is validated and verified, enabling the automatic generation of new code for the revised behavior of the DT with DSLs. This pipeline is set to support the concurrent composition of DTs within SoS.

Future research should prioritize enhancing the pipeline constituents introduced here. Critical advancements include refining the semantic DT model to ensure a standardized, clear description of both physical and virtual elements, specifically necessary DT information within CPSs. Additionally, research should extend to the Composition, Validation, and Generator Engines. For the Composition Engine, exploring effective symbolic reasoning approaches is crucial to facilitate the automated composition of new DT functions. Furthermore, in the Generator Engine, developing associated behavior models using DSLs needs exploration to enable diverse digital model creation, such as simulations and machine learning applications, for various use cases.

Acknowledgment

This research [project ProMoDi and MBDO] is funded by dtec.bw (Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union - NextGenerationEU) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, 505496753)

References

[1] K. Hölldobler, J. Michael *et al.*, “Innovations in model-based software and systems engineering,” *Journal of Object Technology*, vol. 18, no. 1, 2019.

[2] J. Michael, J. Pfeiffer *et al.*, “Integration challenges for Digital Twin Systems-of-Systems,” in *Proceedings of the 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems*, ser. SESoS ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 9–12.

[3] A. Köcher, A. Hayward, and A. Fay, “Model-Based Engineering of CPPS Functions and Code Generation for Skills,” in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2022, pp. 01–08.

[4] M. Hankel and B. Rexroth, “The Reference Architectural Model Industrie 4.0 (Rami 4.0),” *ZVEI*, vol. 2, no. 2, pp. 4–9, 2015.

[5] F. Tao, H. Zhang *et al.*, “Digital Twin in Industry: State-of-the-Art,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[6] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, “Digital Twin in manufacturing: A categorical literature review and classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing 2018.

[7] L. M. Reinpold, L. P. Wagner *et al.*, “Systematic comparison of Software Agents and Digital Twins: Differences, Similarities, and Synergies in Industrial Production,” *Journal of Intelligent Manufacturing*, 2024.

[8] L.-T. Reiche, C. S. Gundlach *et al.*, “The Digital Twin of a System: A Structure for Networks of Digital Twins,” in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2021, pp. 1–8.

[9] M. S. Gill, L.-T. Reiche, and A. Fay, “Method for selecting Digital Twins of Entities in a System-of-Systems approach based on essential Information Attributes,” in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2022, pp. 1–8.

[10] Plattform Industrie 4.0, “Details of the asset administration shell - part 1 the exchange of information between partners in the value chain of industrie 4.0 (version 3.0rc02),” 2022, accessed on 07 June 2024.

[11] C. Hildebrandt, A. Köcher *et al.*, “Ontology Building for Cyber–Physical Systems: Application in the Manufacturing Domain,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1266–1282, 2020.

[12] A. Butting, N. Jansen *et al.*, “Towards Modular Development of Reusable Language Components for Domain-Specific Modeling Languages in the MagicDraw and MontiCore Ecosystems.” *The Journal of Object Technology*, vol. 22, p. 1:1, 01 2023.

[13] S. R. Bader and M. Maleshkova, “The Semantic Asset Administration Shell,” in *Semantic Systems. The Power of AI and Knowledge Graphs*. Springer Nature, 2019, pp. 159–174.

[14] C. Hildebrandt, M. Glawe *et al.*, “Reasoning on engineering knowledge: Applications and desired features,” in *The Semantic Web*, E. Blomqvist, D. Maynard *et al.*, Eds. Cham: Springer International Publishing, 2017, pp. 65–78.

[15] S. Sobernig, B. Hoisl, and M. Strembeck, “Requirements-driven testing of domain-specific core language models using scenarios,” in *2013 13th International Conference on Quality Software*. IEEE, 2013, pp. 163–172.

[16] F. Becker, P. Bibow *et al.*, “A Conceptual Model for Digital Shadows in Industry and Its Application,” in *Conceptual Modeling*, ser. Information Systems and Applications, Cham, 2021, pp. 271–281.

[17] A. Elhabbash, Y. Elkhatib *et al.*, “Principled and automated system of systems composition using an ontological architecture,” *Future Generation Computer Systems*, vol. 157, pp. 499–515, 2024.

[18] L. Bettini, *Implementing Domain-Specific Languages with Xtext and Xtend*. Packt Publishing Ltd, 2016.